JC Luby

Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804

REVERBERATION, NOISE, AND SIGNAL GENERATION ALGORITHM (RENSGEN)

BY: J. E. SENTZ AND J. WAKELEY

Technical Note
File No. 86-64
18 April 1986

Copy No. 8

Department of the Navy
Space and Naval Warfare Systems Command

Contract No. N00024-85-C-6041

Applied Research Laboratory
The Pennsylvania State University
Post Office Box 30
State College, PA 16804

REVERBERATION, NOISE, AND SIGNAL GENERATION ALGORITHM (RENSGEN)

BY: J. E. SENTZ AND J. WAKELEY

Technical Note
File No. 86-64
18 April 1986

Copy No. 8

Department of the Navy
Space and Naval Warfare Systems Command

Contract No. N00024-85-C-6041

Abstract:

The purpose of this study was to develop an Artificial Intelligence Expert System to detect signals in a range-Doppler map of noise and reverberation. This memorandum documents the initial phase of the study, i.e., development of a mathematical representation for a gain-controlled range-Doppler map capable of simulating various in-water acoustic background conditions.

The range-Doppler map representation was generated by an algorithm written in COMMON LISP identified as RENSGEN. RENSGEN was designed to generate a range-Doppler map with options for the presence of signals, noise, and a representation of reverberation assuming a pulsed pure tone transmitted signal. A simulated range-Doppler map of noise and reverberation generated by RENSGEN is shown in Figure 1.

This work was the result of a joint effort by The Applied Research Laboratory and the Mathematics Department of The Pennsylvania State University. Through a Mathematics Work-Study program, they provide a one-year assistantship to University seniors majoring in Mathematics.

April 18, 1986
JES:JW:mdp

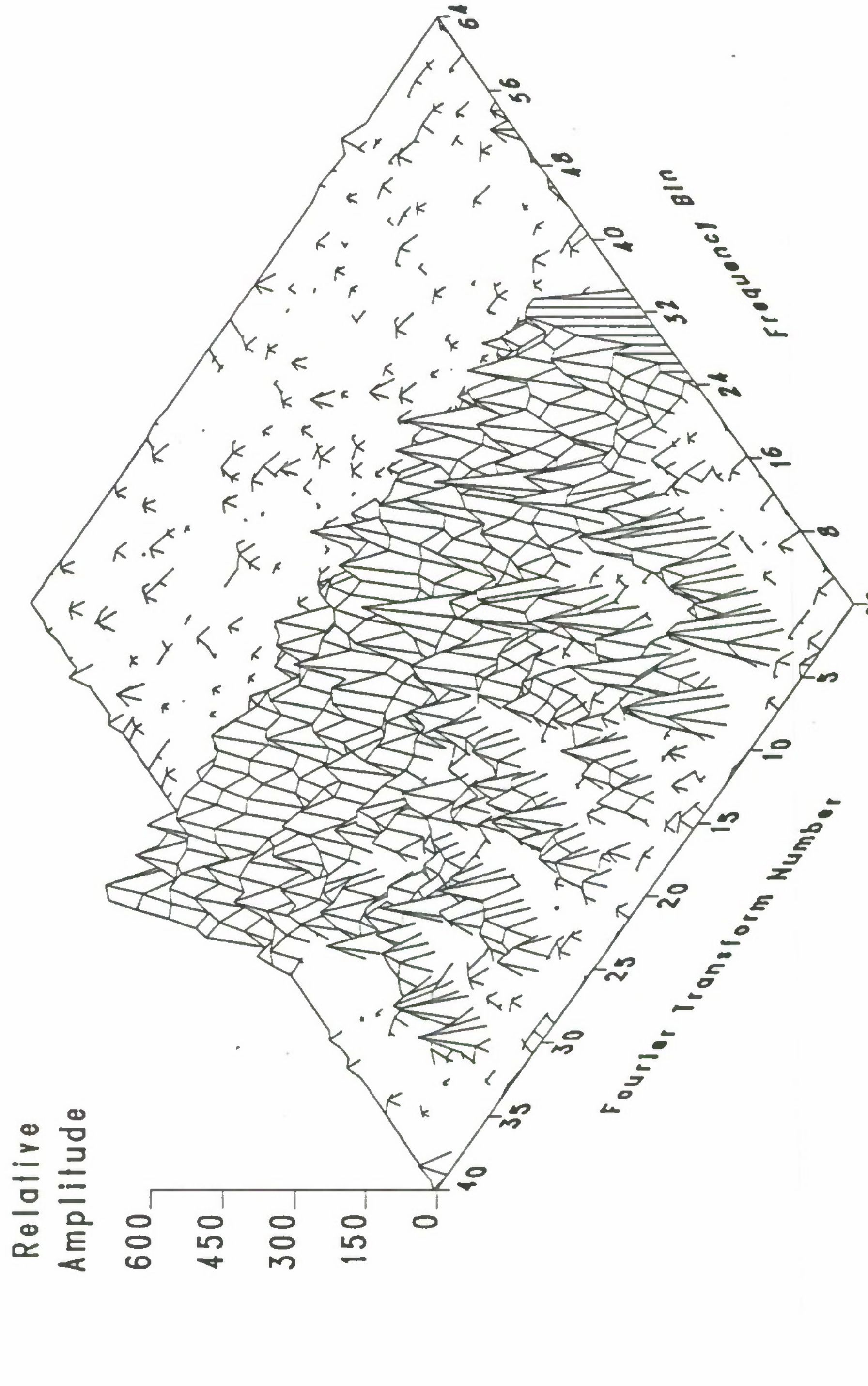


Figure 1. A simulated range-Doppler map of noise and reverberation.

TABLE OF CONTENTS

Page No.

April 18, 1986
JES:JW:mdp

LIST OF FIGURES

INTRODUCTION

This memorandum reports on the first part of a study designed to apply Artificial Intelligence to the detection of acoustic signals in the presence of underwater noise and reverberation environments.

The algorithm this memorandum addresses is called RENSGEN. RENSGEN is an acronym for REverberation, Noise and Signal GENeration. As the name suggests, RENSGEN is concerned with simulating a frequency domain representation of noise and reverberation to be used in the study of signal detection. RENSGEN was designed to simulate the range-Doppler map of the type of measured data shown in Figure 2.

The subsequent portion of the study deals with a program entitled RENSID (REverberation, Noise and Signal IDentification), which attempts to characterize, by amplitude and location, the signals generated in RENSGEN. RENSID will be discussed at a later date.

A range-Doppler map is a three dimensional data presentation of successive Fourier transformed, or fast-Fourier transformed, time domain information to the frequency domain. Frequency is generally expressed as Doppler, relative to the measuring platform, or segmented into frequency bins. Successive Fourier transforms represent increasing time which may be converted to range separation between measuring platform and acoustic reflector. Elapsed time and range separation between source and reflector are related by the speed of sound. Hence the name range-Doppler map.

Figure 2 displays reverberation measured from a moving platform with clearly recognizable surface and bottom returns. Figure 2 is comparable to the approximation generated by RENSGEN in Figure 3. The main difference between Figures 2 and 3 is that frequency (or frequency bin) and the time (or

Figure 2. Measured reverberation

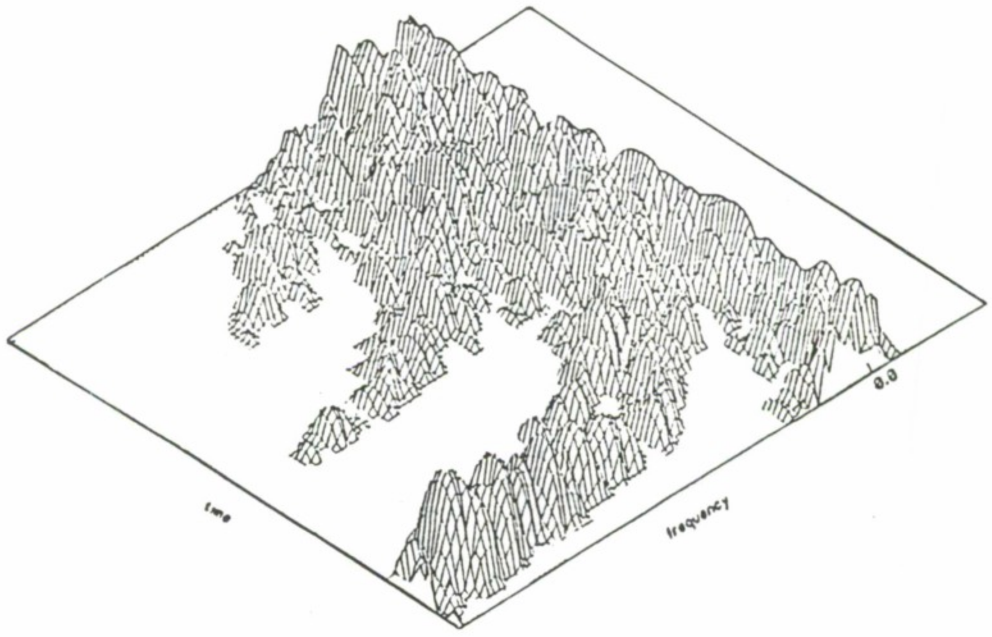
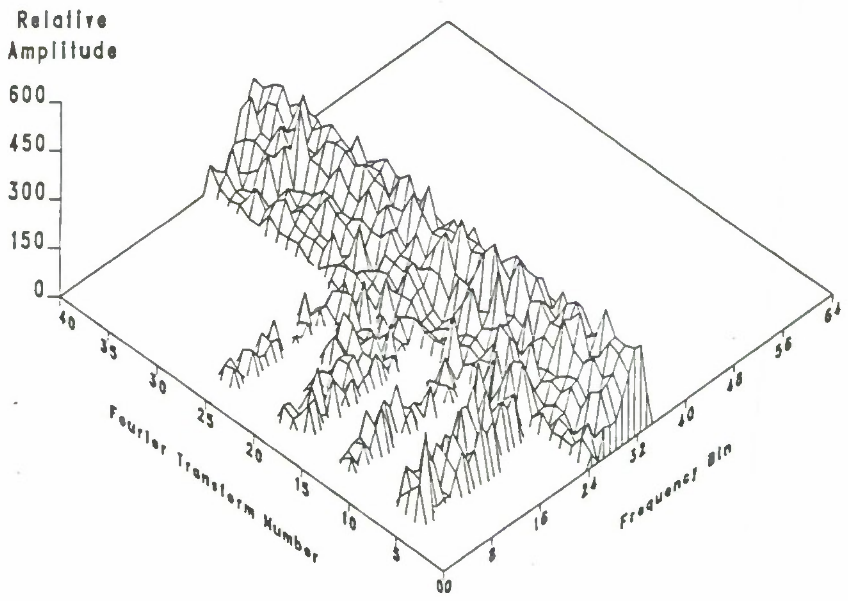


Figure 3. Simulated reverberation



Fourier transform number) scales are greater in Figure 3 by a factor of approximately two.

RENSGEN generates range-Doppler maps by one of two methods. The first method involves random placement of selected signals and optional addition of noise and reverberation environments. The second method uses a regeneration of maps, retaining original map components, but changing the positions of selected signals, i.e., the simulation of reflector motion.

The development of this program was used to study LISP (a language of Artificial Intelligence), and to gain knowledge about the field of Artificial Intelligence (AI). Familiarization with LISP and AI gained during the implementation of this algorithm was applied to the formulation of the subsequent RENSID algorithm.

REVERBERATION, NOISE, AND SIGNAL GENERATION

The REverberation, Noise and Signal GENeration algorithm (hereafter referred to as RENSGEN) generates simulated range-Doppler maps to provide data for a signal detection algorithm entitled RENSID. Each execution of RENSGEN results in one of two basic methods for generating a range-Doppler map. The first method available generates a new range-Doppler map. The second method regenerates a map by retaining previous map background components, i.e., noise and reverberation, and repositioning the simulated signals on the map. The LISP code for the RENSGEN algorithm is presented in Appendix A.

The first method of map generation produces a map from original data. Prior to the production of the map, data specifying the desired map conditions must be supplied. Samples of this input data and the method by which it is attained can be found in Appendix B. The range-Doppler map may include (or exclude) signals, noise, and reverberation. For the purpose of this memorandum reverberation has been divided into two components: (1) boundary returns, i.e., easily distinguishable returns, both single and multiple, from the air-water surface and the bottom capable of being measured with relatively wide angle acoustic beams, and (2) reverberation, i.e., a composite of volume and boundary reverberation not easily separable. The map may contain constant-level or variable-level signals and boundary returns. The number of each type of signal and the mean amplitude levels for all components are input.

The alternative method of map generation is to produce a second-generation map, that is, a map which is regenerated from a previous map. The new map will retain all of the components of the first map (i.e., number and type of signals, presence and level of noise, reverberation, and boundary returns). The difference between the two maps will be in the positioning of the signals.

Whereas the first map contained signals which were all placed on the map randomly by the algorithm, this second-generation repositions selected signals according to a time-velocity relation. Any signals not repostioned in this manner will again be randomly placed on the map. (Note: Random components of the original map, e.g. noise levels, are again random in the second-generation map. However, the distribution of these random levels are centered around the same previously selected mean.)

The RENSGEN algorithm has the capability of placing seven different signal types on a range-Doppler map. These presently available signal types are demonstrated in Figures 4 and 5. The signals selected spell the word LOT, for the person in Genesis who was directed not to look back on the "plains" where he lived, but look forward for "new sources of data." The types are referred to as L, 'fat' L, O, 'fat' O, T, 'fat' T, and bar. Note the portions of the 'bar' and the 'O' that are common in Figure 5 combine. This combination is formed by using the square root of the sum of squares of the amplitudes of the signals involved. Signals can appear on a map in one of two forms; constant-level signals or variable-level signals. For constant-level signals, the amplitude of each cell crossing is equal to the mean crossing level (supplied by the user). Randomly varying signals have cell crossing amplitudes which are Rayleigh distributed about the mean crossing level. Figure 6 shows the constant-level signal configuration of Figure 4 with variable-level signals and no background included.

RENSGEN is capable of including up to three types of background or noise-related components on a range-Doppler map. These components are called 'flat' noise, reverberation (or weighted noise), and boundary returns.

Figure 4. Constant-level signals

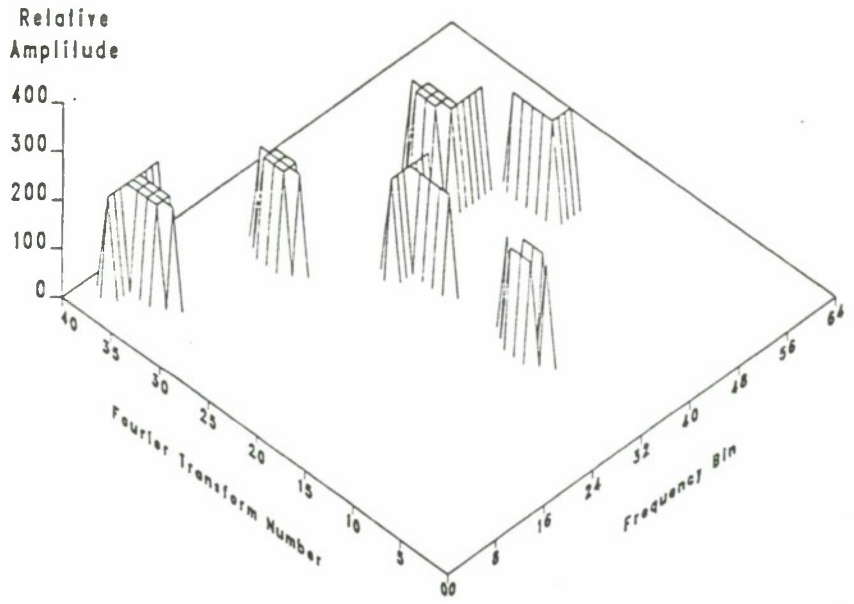


Figure 5. Constant-level signals with bar

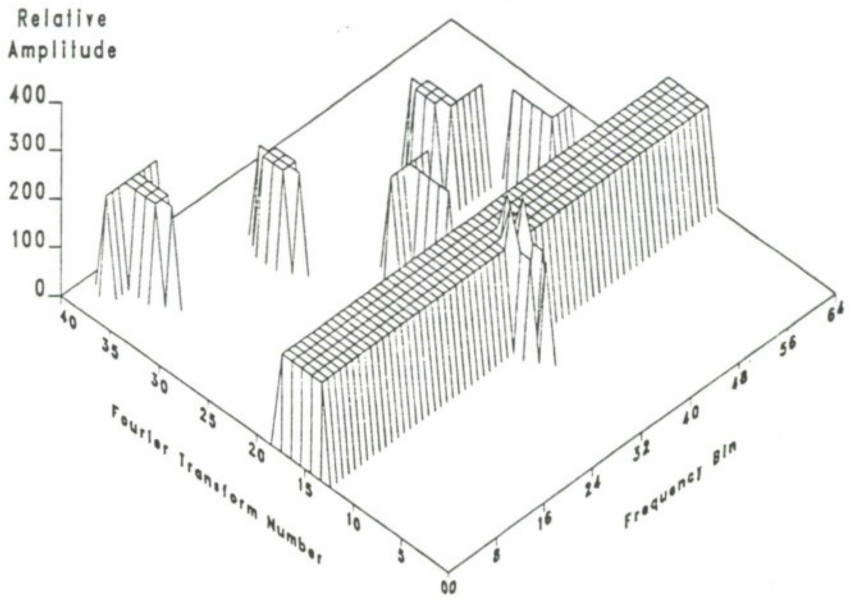


Figure 4. Constant-level signals

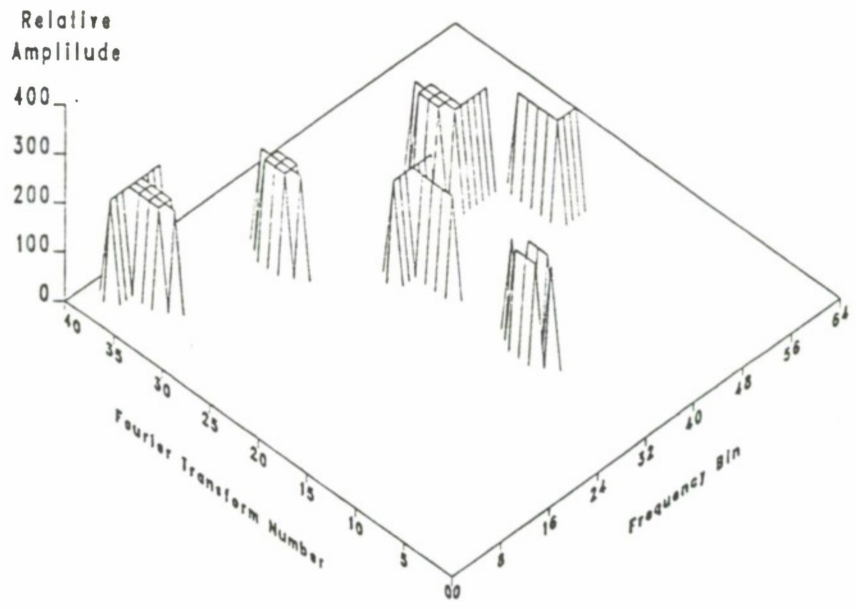
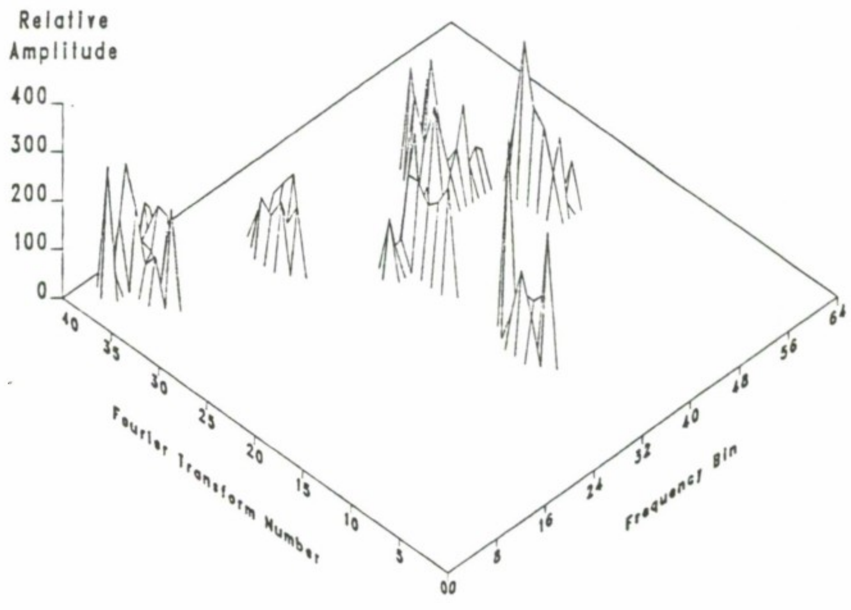


Figure 6. Variable-level signals



'Flat' noise to be added to the map is generated by a Raleigh density function about a user-supplied mean value. An example of a range-Doppler map generated with a high (10:1) mean signal to noise level ratio is shown in Figure 7. Figure 8 shows a map with a low (4:1) mean signal to noise ratio.

Reverberation, or weighted noise, is also generated by a form of the Raleigh distribution. However, the Raleigh density function in this case is altered so that a weighting factor can be introduced. The reverberation is designed to peak at the frequency bin value of 32 which is defined as zero Doppler. Figure 9 shows a map with signals and reverberation. The mean amplitude of the reverberation is supplied by the user. The user also supplies a level for the random component of the reverberation (analogous to 'flat' noise), and a value which determines the shape (width) of the reverberation peak.

Boundary returns can be included on the range-Doppler map in many forms. Boundary return components include returns from the surface, the bottom, and various combinations of surface and bottom (e.g., returns reflected from the surface once and the bottom once (S1B1), the bottom twice and the surface once (B2S1), etc.). An example of a map with all possible boundary returns implemented in RENSGEN is given in Figure 10. Figure 11 demonstrates a map which contains all signals (excluding the bar) and noise-related components that RENSGEN is capable of producing. Figure 12 shows the same data shown in Figure 11 with a 90-degree rotation of the horizontal plane about a vertical axis.

Figure 7. Signals with noise background, high signal to noise ratio

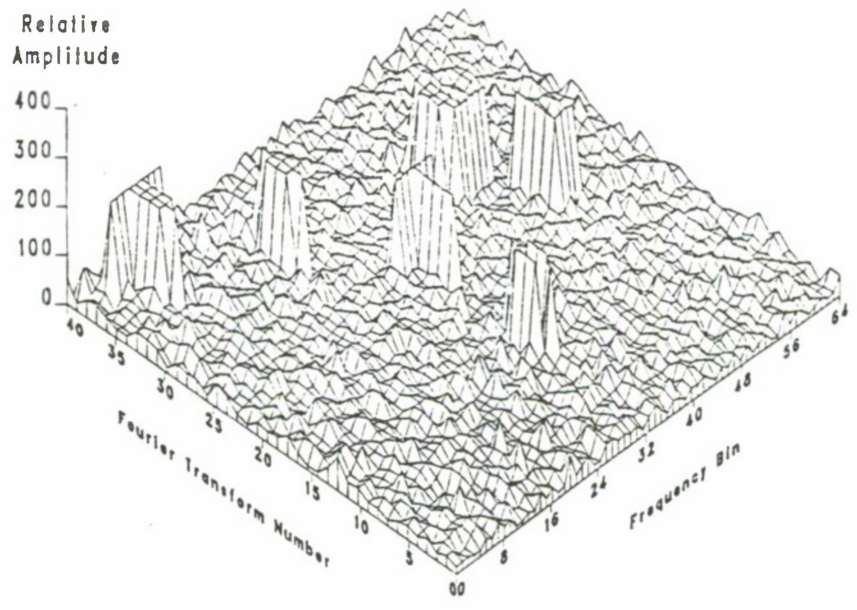


Figure 8. Signals with noise background, low signal to noise ratio

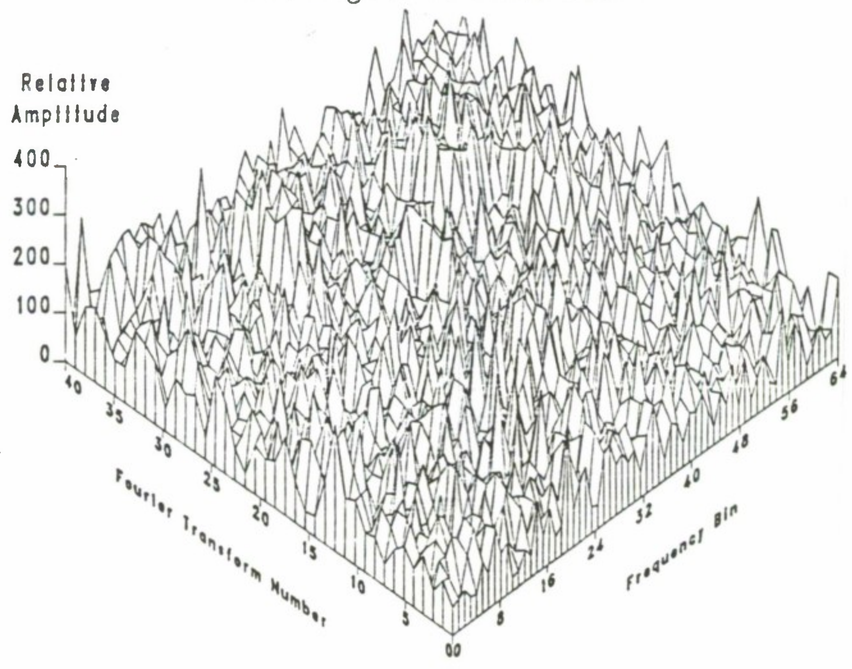


Figure 4. Constant-level signals

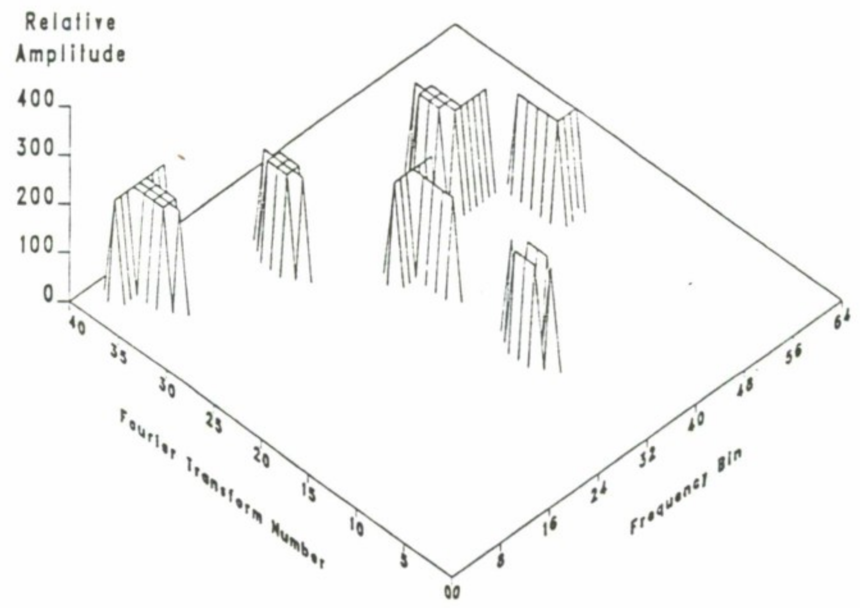


Figure 9. Signals with reverberation

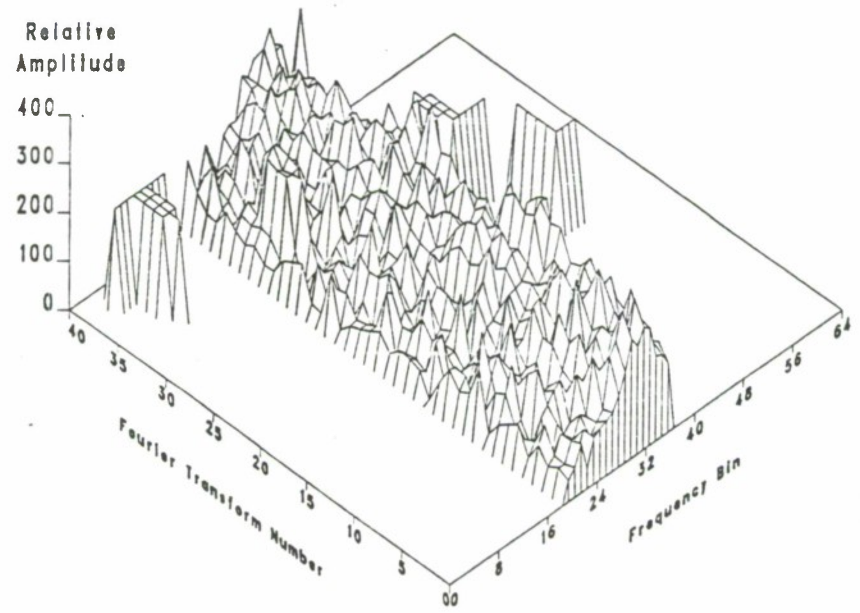


Figure 4. Constant-level signals

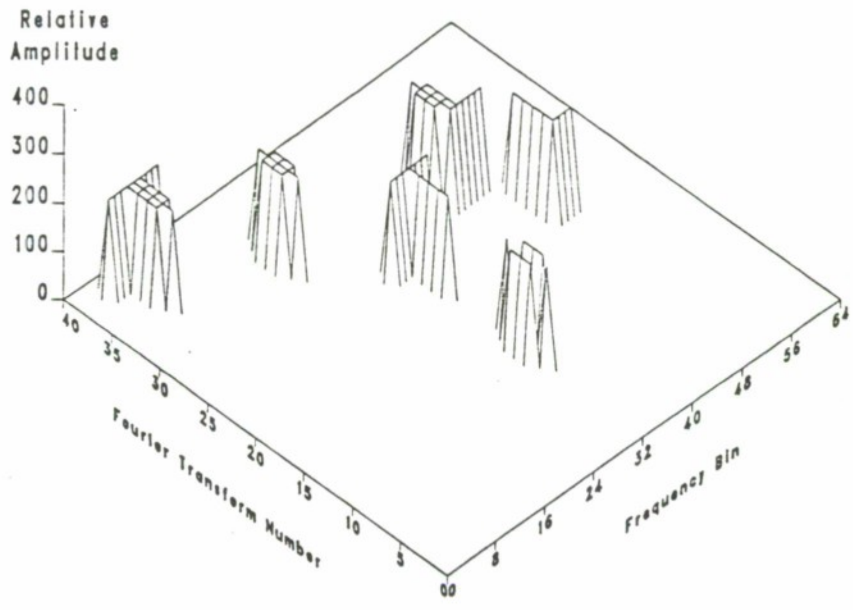


Figure 10. Signals with reverberation and boundary returns

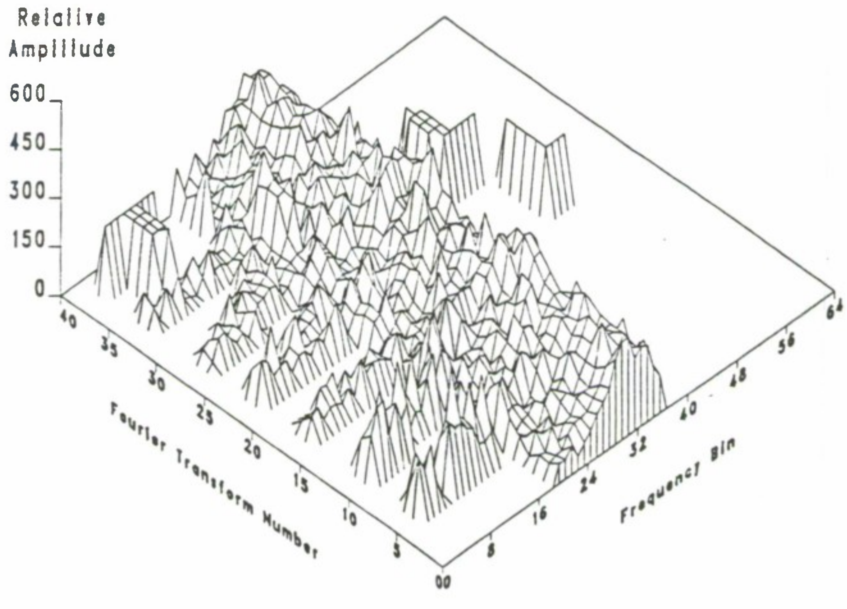


Figure 11. Signals, noise, reverberation and boundary returns

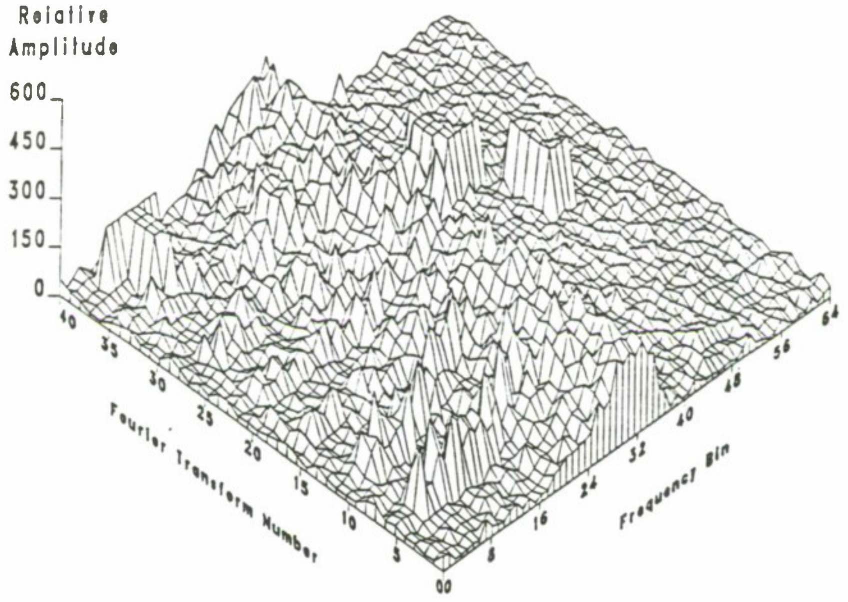
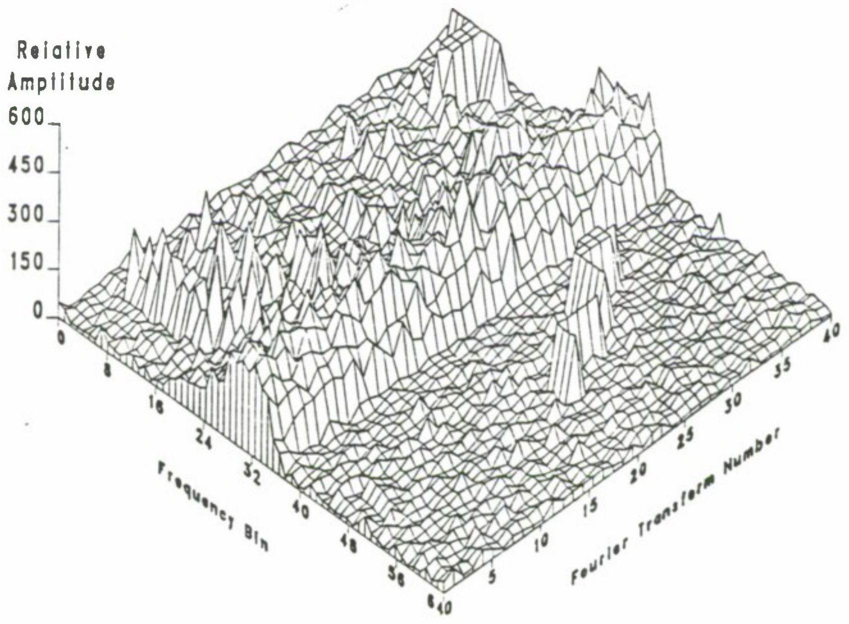


Figure 12. A 90-degree rotation of Figure 11



A second-generation map is produced by regenerating a previous map with the same basic components. The main objective in the second-generation process is to estimate the change in position of the signals after a fixed time interval. The positional change is calculated assumming straight line motion by the acoustic reflector located directly in front of the measuring platform. Signals may also be repositioned in order to eliminate overlapping and to separate from excessive interference. The second map retains the number and type of signals in the original map as well as levels of signals, noise, reverberation and boundary returns whenever appropriate. The regeneration process does, however, generate unique values for individual map cells while still obeying the same random distribution functions of the original method. The second-generation map shown in Figure 13 was regenerated using Figure 4 as the previous map. In this example, the 'fat' L, 'fat' T, 'fat' O and L signals were repositioned. (Note that the 'fat' T is no longer visible on the map, having moved beyond the upper limit of the scale.)

RENSGEN is run in an interactive LISP environment. Sample interactive terminal sessions are presented in Appendix B. The samples are records of actual data used in generation of Figure 11 and Figure 13 (Method 1 and 2, respectively). RENSGEN creates four output files during each execution, and uses one input file when the second-generation method is selected.

An input file, CENTERS.LSP, is used when a specific map is to be regenerated. This file contains all of the data input to the program during the original generation as well as identification and location of each signal present on the map.

Figure 4. Constant-level signals

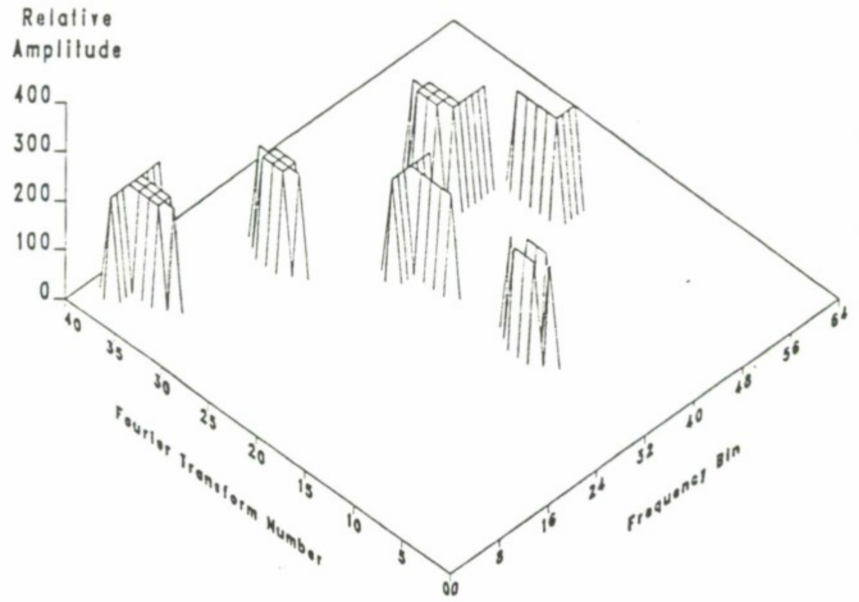
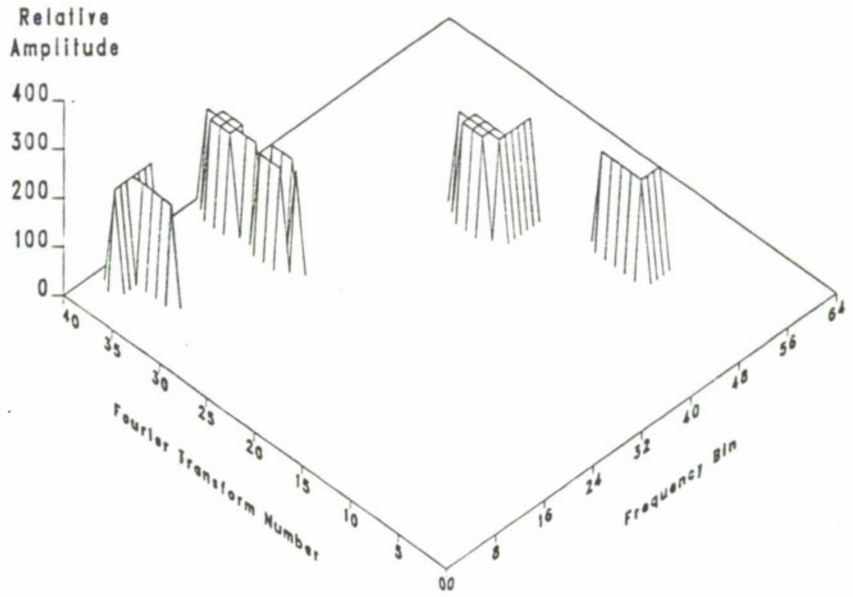


Figure 13. Repositioned signals



RENSGEN creates three files for a quick look at the resulting output, SIGNALS.LSP, READIN.LSP, and CENTERS.LSP (See Appendix C). A fourth file, PLOT3D.LSP, is also created by RENSGEN for the purpose of computer aided plotting. The file stored in SIGNALS.LSP presents either a quantized representation of the signal and/or noise level values or a representation that shows only the presence of a signal. The files stored in READIN.LSP and PLOT3D.LSP present the actual signal and/or noise level values. The READIN.LSP file is to be read and used in the RENSID.LSP program, designed to identify the signals. The PLOT3D.LSP file is to be read and used in the RDPLOT.FOR program to produce three-dimensional graphs of the range-Doppler map, Appendix D. The file stored in CENTERS.LSP keeps a record of the information input into the program to be used by the program if a certain map is to be regenerated with specified signals repositioned.

Additional information concerning the implementation and use of RENSGEN is included in Appendices E, F, G, and H. Appendix E provides instructions for using the VAX/LISP. Appendix F outlines some interesting LISP functions used in the implementation of RENSGEN. Appendix G lists the input data used in generation of Figures 1 through 13. Appendix H contains references used in writing RENSGEN.

APPENDIX A

RENSGEN LISP CODE

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                                        APPENDIX A
;
;    LISP Program:  RENSGEN.LSP
;
;    AUTHDRS: Joe Wakeley, ARL/PSU
;                 and
;             Jodi E. Sentz, MATH/PSU
;
;    REVISED: 27 February 1986
;
;
;    REFERENCES: (1) Wilensky, Robert, "LISPCRAFT," University of
;                    California, Berkely, W.W. Norton & Company, 1984.
;
;                (2) Winston, Patrick H. and Born, Berthold K. P.,
;                    "LISP," Massachusetts Institute of Techology,
;                    Addison-Wesley Publishing Company, 1981.
;
;                (3) Steele, Guy L., "Common LISP - The Language,"
;                    Digital Press, 1984.
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN REVERBERATIDN ()                                                          ; FUNCTIDN TD READ IN DATA, INITIALIZE,
;                                                                                ;   RUN RENSGEN, AND SET UP DUTPUT FILE
;
      (SETQ LDOK (DPEN "SIGNALS.LSP;1" :DIRECTION :OUTPUT :IF-EXISTS :NEW-VERSIDN))    ; OPEN OUTPUT FILE LOOK FDR
;                                                                                ;   QUANTIZED SIGNAL VALUES
      (SETQ REVIDEN (DPEN "READIN.LSP;1" :DIRECTION :DUTPUT :IF-EXISTS :NEW-VERSIDN))  ; DPEN OUTPUT FILE REVIDEN FDR
;                                                                                ;   SIGNAL VALUES TD BE USED IN REVID
      (SETQ PLDT3D (DPEN "PLDT3D.LSP;1" :DIRECTIDN :DUTPUT :IF-EXISTS :NEW-VERSIDN))   ; DPEN DUTPUT FILE PLDT3D FDR PLDTTING
;                                                                                ;   SIGNAL VALUES WITB REVPLOT
;
      (WRITE-CBAR #\NEWLINE LDDK)        (WRITE-CBAR #\NEWLINE LDDK)                ; PUT BLANK LINES IN DUTPUT FILE LODK
      (WRITE-CBAR #\NEWLINE LDDK)        (WRITE-CBAR #\NEWLINE LOOK)                ;   SD DUTPUT WILL BE CENTERED
      (WRITE-CBAR #\NEWLINE LOOK)        (WRITE-CBAR #\NEWLINE LDDK)                ;   DN PAGE
      (WRITE-CBAR #\NEWLINE LOOK)        (WRITE-CBAR #\NEWLINE LOOK)
;
      (RESET)                                                                    ; CALL FUNCTIDN TD INITIAITE VARIABLES
;
      (INITIAL)                                                                  ; CALL FUNCTIDN TQ INITIALIZE IN_LINE_FT
;
      (TERPRI)
      (PRINC "Would you like a new (N) range-Doppler map or")                   ; PRDMPT TD SET TYPE OF MAP
      (PRINC " will this be a second (S) generation map? [N/S]  ")
;
      (COND ((EQUAL (READ) 'S) (REGEN))                                          ; IF REPEAT, REGENERATE MAP
            (T                                                                   ;         DR
              (PRDMPT)))                                                         ; IF NEW, PRDVIDE PROMPTS
;
      (WRITE_VALUE "SIGNAL MEAN :   " SIGS_PLACE SIG_MU)                         ; WRITE SIGNAL MEAN TO FILE REVIDEN
      (WRITE_VALUE "REVERBERATION MEAN LEVEL :   " REV_WEIGHT_NDISE REV_WN_MU)   ; WRITE REVERB MEAN LEVEL TD FILE REVIDEN
      (WRITE_VALUE "FLAT NDISE MEAN :   " FLAT_NDISE FN_MU)                      ; WRITE FLAT NDISE MEAN TD FILE REVIDEN
;
      (COND ((EQUAL QUANT 1)
;
            (WRITE_Q_VALUE "QUANTIZED SIGNAL MEAN :   " SIGS_PLACE SIG_MU)        ; WRITE QUANTIZED SIGNAL MEAN
;                                                                                ;  TO FILE LDDK
;
            (WRITE_Q_VALUE "QUANTIZED REVERBERATIDN MEAN LEVEL:   "
                                                    REV_WEIGHT_NDISE REV_WN_MU)  ; WRITE QUANTIZED REVERBERATIDN
;                                                                                ;  MEAN LEVEL TD FILE LOOK
;
            (WRITE_Q_VALUE "QUANTIZED FLAT NDISE MEAN :   " FLAT_NDISE FN_MU)))   ; WRITE QUANTIZED WEIGRTED NDISE
;                                                                                ;  MEAN TQ FILE LDDK
;
      (SET_SIGS SET_NUM_FL SET_NUM_FD SET_NUM_FT                                 ; CALL SET_SIGS FUNCTION TO
                SET_NUM_L SET_NUM_O SET_NUM_T SET_NUM_B)                         ;   EXECUTE PROGRAM
;
      (BDUND_WRITE MIDDLE BDUND_GEN)                                             ; CALL BOUND_WRITE FUNCTIDN TO
;                                                                                ;   WRITE TD FILE MIDDLE
      (WRITE CEN_SIG :STREAM MIDDLE)                                             ; WRITE CEN_SIG TD FILE MIDDLE
;
      (CLDSE LDDK)                                                               ; CLDSE DUTPUT FILE LOOK
      (CLDSE REVIDEN)                                                            ; CLDSE DUTPUT FILE REVIDEN
      (CLDSE MIDDLE)                                                             ; CLOSE DUTPUT FILE MIDDLE (DPENED IN REGEN)
      (CLOSE PLDT3D))                                                            ; CLDSE OUTPUT FILE PLDT3D
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                            INITIAL CONDITIONS I
;
(DEFUN INITIAL ()                                                         ; FUNCTION TO ADDRESS 40 CONSECUTIVE 64 POINT
;                                                                         ;   FOURIER TRANSFORMS CALLED IN_LINE_FT
     (PROG (GENER KOUNT COUNT)
;
           (SETQ GENER ())
           (SETQ KOUNT 0)
           (SETQ COUNT 0)
           (SETQ IN_LINE_FT ())
;
         LOOP1                                                            ; LOOP TO SET UP ROW NUMBERS
;
            (SETQ COUNT 0)                                                ; RESET COLUMN COUNTER
            (SETQ KOUNT (+ KOUNT 1))                                      ; INCREMENT ROW COUNTER
            (SETQ ROS (* KOUNT 100))                                      ; SET ROS TO 100 * NUMBER OF ROWS
            (GENSYM ROS)                                                  ; START GENERATOR AT VALUE OF ROS
;
            (COND ((> KOUNT 40)                                           ; IF ALL 40 ROWS GENERATED,
                   (SETQ IN_LINE_FT (REVERSE IN_LINE_FT))                 ;    REVERSE IN_LINE_FT
                   (RETURN IN_LINE_FT))                                   ;    RETURN
                  (T (GO LOOP2)))                                         ;  OTHERWISE, CONTINUE AT LOOP2
;
           LOOP2
;
               (SETQ COUNT (+ COUNT 1))                                   ; INCREMENT COLUMN COUNTER
               (SETQ GENER (GENSYM "C"))                                  ; GENERATE A VALUE WITH THE PREFIX "C"
               (SETQ IN_LINE_FT (CONS GENER IN_LINE_FT))                  ; ADD THE VALUE TO IN_LINE_FT
;
                (COND ((> COUNT 63 ) (GO LOOP1))                          ; IF ALL 64 ARE DONE, GO TO LOOP1
                      (T (GO LOOP2)))))                                   ;    IF NOT, REPEAT
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               INITIAL CONDITIONS II
;
(DEFUN RESET ()                                                             ; FUNCTION TO RESET VARIABLES TO RUN
;                                                                           ;   THE PROGRAM
;
      (SETQ SIG_FL '())                                                     ; 'FAT' L SIGNALS
      (SETQ SIG_FO '())                                                     ; 'FAT' O SIGNALS
      (SETQ SIG_FT '())                                                     ; 'FAT' T SIGNALS
      (SETQ SIG_L '())                                                      ; L SIGNALS
      (SETQ SIG_O '())                                                      ; O SIGNALS
      (SETQ SIG_T '())                                                      ; T SIGNALS
      (SETQ SIG_B '())                                                      ; B SIGNALS
;
      (SETQ LOT_SIGS '())                                                   ; ALL SIGNALS
      (SETQ ADD_SIGS '())                                                   ; ADDITIONAL SIGNALS REPOSITIONED
;
      (SETQ CEN_SIG '())                                                    ; LIST OF SIGNAL SHAPES AND CENTERS
;
      (SETQ MID_SIG '())                                                    ; NEWLY GENERATED SIGNAL AND CENTER
      (SETQ NUMBER '1)                                                      ; NUMBER OF SIGNALS TO REPOSITION
;
      (SETQ YES_STARS 9)                                                    ; ELEMENT FOR OUTPUT FILE
      (SETQ NO_STARS 1)                                                     ; ELEMENT FOR OUTPUT FILE
;
      (SETQ OUTPUT_LIST ())                                                 ; OUTPUT LIST FOR FILE LOOK
      (SETQ OUTPUT_FILE ())                                                 ; OUTPUT LIST FOR FILE REVIDEN
      (SETQ OUTPUT_FILE1 ())                                                ; OUTPUT LIST FIR FILE PLOT3D
;
      (SETQ COUNT 0)                                                        ; COUNTER FOR OUTPUT FILE LOOK
      (SETQ KOUNT 0)                                                        ; COUNTER FOR OUTPUT FILE REVIDEN
;
      (SETQ SET_NUM_FL 0)                                                   ; VARIABLES TO READ IN
      (SETQ SET_NUM_FO 0)                                                   ;   DESIRED NUMBER OF SIGNALS
      (SETQ SET_NUM_FT 0)                                                   ;   OF EACB TYPE
      (SETQ SET_NUM_L 0)
      (SETQ SET_NUM_O 0)
      (SETQ SET_NUM_T 0)
      (SETQ SET_NUM_B 0)
;
      (SETQ SIGS_PLACE 0)                                                   ; FLAG TO PLACE SIGNALS
      (SETQ REV_WEIGRT_NOISE 0)                                             ; FLAG TO PLACE WEIGHTED NOISE
      (SETQ FLAT_NOISE 0)                                                   ; FLAG TO PLACE NOISE
      (SETQ BOUND_FLAG 0)                                                   ; FLAG TO PLACE BOUNDARY CONDITIONS
;
      (SETQ SIG_MU 0)                                                       ; VALUE FOR SIGNAL MEAN
      (SETQ REV_WN_MU 0)                                                    ; VALUE FOR REVERB WEIGRTED NOISE MEAN
      (SETQ REV_ALPBA 0)                                                    ; VALUE FOR SBAPE OF REVERB WEIGBT NOISE
      (SETQ REV_ALPBA2 0)                                                   ; VALUE FOR REVERB RANDOM COMPONENT
      (SETQ FN_MU 0)                                                        ; VALUE FOR FLAT NOISE MEAN
      (SETQ BD_VAL_SQ 0)                                                    ; VALUE FOR BOUNDARY RETURN
;
      (SETQ QUANT 0))                                                       ; FLAG TO QUANTIZE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN WRITE_VALUE (TO_WRITE ID_FLAG XVALUE)                                            ; FUNCTION TO WRITE HEADING AND VALUES
                                                                                        ;   TO FILE REVIDEN
;
       (WRITE TO_WRITE :STREAM REVIDEN)                                                 ; WRITE HEADING TO REVIDEN
;
       (COND ((EQUAL ID_FLAG 1)                                                         ; WRITE VALUE TO REVIDEN
              (WRITE XVALUE :STREAM REVIDEN))
;
             (T
              (WRITE '0 :STREAM REVIDEN)))
;
       (WRITE-CHAR #\NEWLINE REVIDEN))                                                  ; SKIP A LINE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN WRITE_Q_VALUE (TO_WRITE ID_FLAG VALUE_MU)                                        ; FUNCTION TO WRITE HEADING AND VALUES
;                                                                                       ;   TO FILE LOOK
             (COND ((EQUAL ID_FLAG 1) (WRITE TO_WRITE :STREAM LOOK)
                                      (SETQ QXVALUE (+ (TRUNCATE (/ VALUE_MU 32)) 1))
                                      (WRITE QXVALUE :STREAM LOOK)
                                      (WRITE-CHAR #\NEWLINE LOOK))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN REGEN ()                                                                         ; FUNCTION TO REGENERATE SIGNALS FROM
;                                                                                       ;   ANOTHER MAP
       (MAKE-PATHNAME :VERSION :NEWEST)  (SETQ INPUT (OPEN "CENTERS.LSP"))              ; READ DATA FROM FILE CONSTRUCTED IN PROMPT
       (SETQ SIGS_PLACE (READ INPUT))                                                   ;   FUNCTION DURING PREVIOUS RUN OF PROGRAM
       (COND ((EQUAL SIGS_PLACE 1)
              (SETQ SET_NUM_FL    (READ INPUT))
              (SETQ SET_NUM_FO    (READ INPUT))
              (SETQ SET_NUM_FT    (READ INPUT))
              (SETQ SET_NUM_L     (READ INPUT))
              (SETQ SET_NUM_O     (READ INPUT))
              (SETQ SET_NUM_T     (READ INPUT))
              (SETQ SET_NUM_H     (READ INPUT))
              (SETQ SIG_MU        (READ INPUT))
              (SETQ RAN_SIG_LEV   (READ INPUT))))
       (SETQ FLAT_NOISE         (READ INPUT))
       (SETQ FN_MU              (READ INPUT))
       (SETQ REV_WEIGHT_NOISE   (READ INPUT))
       (SETQ REV_WN_MU          (READ INPUT))
       (SETQ REV_ALPHA          (READ INPUT))
       (SETQ REV_ALPHA2         (READ INPUT))
       (SETQ QUANT              (READ INPUT))
       (BOUND_READ INPUT)                                                               ; CALL BOUND_READ TO PROMPT FOR
       (SETQ CEN_SIG            (READ INPUT))                                           ;   BOUNDARY RETURNS
       (CLOSE INPUT)                                                                    ; CLOSE FILE FOR INPUT
;
       (SETQ MIDDLE (OPEN "CENTERS.LSP;1" :DIRECTION :OUTPUT :IF-EXISTS :NEW-VERSION))  ; OPEN OUTPUT FILE MIDDLE TO STORE
;                                                                                       ;   REGENERATION INFORMATION
       (WRITE SIGS_PLACE          :STREAM MIDDLE)   (WRITE-CHAR #\NEWLINE MIDDLE)
       (COND ((EQUAL SIGS_PLACE 1)
              (WRITE SET_NUM_FL   :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)        ; WRITE DATA BACK INTO FILE
              (WRITE SET_NUM_FO   :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SET_NUM_FT   :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SET_NUM_L    :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SET_NUM_O    :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SET_NUM_T    :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SET_NUM_H    :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE SIG_MU       :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
              (WRITE RAN_SIG_LEV  :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)))
       (WRITE FLAT_NOISE          :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE FN_MU               :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE REV_WEIGHT_NOISE    :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE REV_WN_MU           :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE REV_ALPHA           :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE REV_ALPHA2          :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
       (WRITE QUANT               :STREAM MIDDLE)  (WRITE-CHAR #\NEWLINE MIDDLE)
;
       (REPOSITION_SIGNAL_PROMPT))                                                      ; CALL FUNCTION TO PROMPT FOR REPOSITIONING
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(OEFUN REPOSITION_SIGNAL_PROMPT ()                                        ; FUNCTION TO PROMPT FOR
;                                                                          ;   SIGNALS TO REPOSITION
       (CONO ((EQUAL SIGS_PLACE 1)
               (PRINC "The last range Ooppler map had these signals and center values: ")
               (TERPRI)
               (PRINC CEN_SIG)
               (TERPRI)
               (PRINC "How many of these signals are to be repositioned?  ")    ; PROMPT TO REAO NUMBER OF SIGNALS TO
               (SETQ NUMBER (REAO))                                          ;   BE REPOSITIONEO
;
       (GENERATE NUMBER)))                                                ; CALL TO FUNCTION GENERATE
       (SET 'CEN_SIG MIO_SIG))                                            ; AOO REGENERATED SIGNALS ANO CENTER VALUES
;                                                                          ;   TO CEN_SIG
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN GENERATE (NUMB)                                                   ; FUNCTION TO GENERATE NEW POSITIONS
;                                                                         ;   FOR SELECTEO SIGNALS
    (PROG (ANS)
;
            LOOP
;
               (COND ((EOUAL NUMB 0) (RETURN NUMB))                      ; WHEN ALL SIGNALS REPOSITIONEO, QUIT
;
                     (T
;
                      (SETQ PARAM  (ASK_REGEN))                          ; SET UP GEOMETRY RELATIVE TO PREVIOUS
                                                                         ;   RANGE OOPPLER MAP ANO RETURN OESIRED
                                                                         ;   SIGNAL TO REPOSITION ANO TIME CHANGE
                      (SETQ ANS (CAR PARAM))                             ; GET ANSWERS FROM REGEN
                      (SETQ TIME_CHANGE (CDR PARAM))                     ;   QUERY TO REPOSITION SIGNALS
;
                       (CONO ((EQUAL (CAR ANS) 'FL)                      ; IF SIGNAL TO BE REPOSITIONED IS A 'FAT' L,
                              (SETQ SET_NUM_FL (- SET_NUM_FL 1))         ;   TAKE 1 FROM FL COUNTER
                              (UP_OATE SET_NUM_FL 'PLACE_SIG_FL 'FL))    ;   CALL UPOATE FUNCTION TO AOD NEW SIGNAL
;
                             ((EQUAL (CAR ANS) 'FO)                      ; IF SIGNAL TO BE REPOSITIONEO IS A 'FAT' O,
                              (SETO SET_NUM_FO (- SET_NUM_FO 1))         ;   TAKE 1 FROM FO COUNTER
                              (UP_DATE SET_NUM_FO 'PLACE_SIG_FO 'FO))    ;   CALL UPOATE FUNCTION TO AOD NEW SIGNAL
;
                             ((EQUAL (CAR ANS) 'FT)                      ; IF SIGNAL TO BE REPOSITIONEO IS A 'FAT' T,
                              (SETQ SET_NUM_FT (- SET_NUM_FT 1))         ;   TAKE 1 FROM FT COUNTER
                              (UP_DATE SET_NUM_FT 'PLACE_SIG_FT 'FT))    ;   CALL UPDATE FUNCTION TO AOO NEW SIGNAL
;
                             ((EQUAL (CAR ANS) 'L)                       ; IF SIGNAL TO BE REPOSITIONED IS AN L,
                              (SETO SET_NUM_L (- SET_NUM_L 1))           ;   TAKE 1 FROM L COUNTER
                              (UP_DATE SET_NUM_L 'PLACE_SIG_L 'L))       ;   CALL UPOATE FUNCTION TO ADD NEW SIGNAL
;
                             ((EQUAL (CAR ANS) 'O)                       ; IF SIGNAL TO BE REPOSITIONEO IS AN O,
                              (SETQ SET_NUM_O (- SET_NUM_O 1))           ;   TAKE 1 FROM O COUNTER
                              (UP_DATE SET_NUM_O 'PLACE_SIG_O 'O))       ;   CALL UPOATE FUNCTION TO AOD NEW SIGNAL
;
                             ((EOUAL (CAR ANS) 'T)                       ; IF SIGNAL TO BE REPOSITIONEO IS A T,
                              (SETQ SET_NUM_T (- SET_NUM_T 1))           ;   TAKE 1 FROM T COUNTER
                              (UP_OATE SET_NUM_T 'PLACE_SIG_T 'T))       ;   CALL UPOATE FUNCTION TO ADO NEW SIGNAL
;
                             ((EQUAL (CAR ANS) 'B)                       ; IF SIGNAL TO BE REPOSITIONEO IS A BAR,
                              (SETQ SET_NUM_B (- SET_NUM_B 1))           ;   TAKE 1 FROM B COUNTER
                              (UP_DATE SET_NUM_B 'PLACE_SIG_B 'B))       ;   CALL UPDATE FUNCTION TO AOO NEW SIGNAL
;
                             (T                                          ; IF SIGNAL TO BE REPOSITIONEO IS INCORRECT,
                               (SETQ NEW ())                             ;   SET NEW TO NIL
                               (SETO NUMB (+ NUMB 1))))                  ;   AOJUST COUNTER TO ITERATE AGAIN
;
                       (SETQ MIO_SIG (CONS NEW MID_SIG))                 ; AOO NEW SIGNAL AND CENTER VALUE
                       (SETO NUMB (- NUMB 1))                            ; OECREASE NUMBER OF SIGNALS TO REPOSITION
;
            (GO LOOP)))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(OEFUN ASK_REGEN ()                                                              ; FUNCTION TO PROMPT FOR SIGNAL ANO TIME
;                                                                                ;   CHANGE, ANO FIND NEW COORDINATES
                    (TERPRI)
                    (PRINC " Enter a signal to reposition,")
                    (PRINC " as it appears above.  ")                            ; PROMPT FOR SIGNAL TO BE REPOSITIONEO
                    (TERPRI)
                    (SETQ ANS (REAO))                                            ; REAO ANSWER
                    (PRINC " Enter the desired time change (sec).")              ; ENTER ELAPSEO TIME SINCE PRECEEOING
                                                                                 ;    RANGE OOPPLER MAP
                    (SETQ TIME_CHANGE (REAO))
;
                    (SETQ CENTER (CADR ANS))                                     ; SET CENTER VALUE
;
                    (SETD Y (TRUNCATE (/ CENTER 64)))                            ; SET Y COORDINATE
                    (SETD X (- CENTER (* Y 64)))                                 ; SET X COOROINATE
;
                    (SETQ Y (- Y (TRUNCATE (/ (* TIME_CHANGE (- X 32)) 18.75))))
;                                                                                ; COMPUTE NEW Y COORDINATE
                    (SETD NEW_CENTER (+ (* 64 Y) X))                             ; COMPUTE NEW CENTER FOR SIGNALS
                    (SETD NEW_CENTER_B (TRUNCATE (/ NEW_CENTER 64)))             ; COMPUTE NEW CENTER FOR BAR
                    (CONS ANS TIME_CHANGE))                                      ; RETURN VALUES OF ANS ANO TIME_CHANGE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN UP_OATE (SET_NUM_ID PLACE_SIG_ID ID)                                        ; FUNCTION TO ADO REPOSITIONEO SIGNALS
;
            (CDND ((EQUAL ID 'B)                                                   ; AOJUSTMENT FOR BAR SIGNAL
                   (SETQ ID_CENTER NEW_CENTER_B))
                  (T
                   (SETQ IO_CENTER NEW_CENTER)))
;
            (SETQ AOO_SIGS (APPEND (FUNCALL PLACE_SIG_ID IO_CENTER) AOO_SIGS))      ; AOD NEW SIGNAL TO AOD_SIGS
            (SETD NEW (CONS ID (CONS IO_CENTER '())))))                            ; ADD LISTING OF NEW SIGNAL TO NEW
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PROMPT ()                                                                  ; FUNCTION TO PROMPT FOR OESIREO TYPE DF MAP
;
      (SETQ MIDOLE (DPEN "CENTERS.LSP;1" :OIRECTION :OUTPUT :IF-EXISTS :NEW-VERSION))   ; OPEN OUTPUT FILE MIDDLE
;
      (TERPRI)
      (PRINC "Are signals desired on the range Doppler map? [Y/N]  ")             ; PROMPT TD INCLUOE SIGNALS
;
      (CONO ((EQUAL 'Y (REAO)) (ASK_SIG))                                         ; IF SIGNALS OESIREO, CALL ASK
            (T
             (WRITE SIGS_PLACE :STREAM MIOOLE)  (WRITE-CHAR #\NEWLINE MIOOLE)))    ; WRITE SIGS_PLACE TO FILE MIOOLE
;
      (TERPRI)
      (PRINC "Is noise desired on the range Ooppler map? [Y/N]  ")                ; PROMPT TO INCLUOE NOISE
;
      (CONO ((EQUAL 'Y (REAO)) (SETQ FLAT_NOISE 1) (ASK_N))
            (T
             (WRITE FLAT_NOISE :STREAM MIOOLE)  (WRITE-CHAR #\NEWLINE MIDDLE)      ; WRITE FLAT NOISE FLAG TO FILE MIDOLE
;
             (WRITE FN_MU :STREAM MIOOLE)  (WRITE-CHAR #\NEWLINE MIOOLE)))         ; WRITE FLAT NOISE MEAN TO FILE MIOOLE
;
      (TERPRI)
      (PRINC "Is reverberation desired on the range Oppler map? [Y/N)  ")        ; PROMPT TO INCLUDE REVERBERATION
;
      (CDND ((EQUAL 'Y (READ)) (SETQ REV_WEIGHT_NDISE 1) (ASK_REV))
            (T
             (WRITE REV_WEIGHT_NOISE :STREAM MIODLE)  (WRITE-CHAR #\NEWLINE MIOOLE)  ; WRITE REVERB WEIGHTEO NOISE FLAG TO
;                                                                                  ; FILE MIDDLE
             (WRITE REV_WN_MU :STREAM MIDOLE)  (WRITE-CHAR #\NEWLINE MIDOLE)       ; WRITE REVERB WEIGHTED NOISE MEAN TO
;                                                                                  ; FILE MIOOLE
             (WRITE REV_ALPHA :STREAM MIODLE)  (WRITE-CHAR #\NEWLINE MIODLE)       ; WRITE REVERB ALPHA TO FILE MIDDLE
;
             (WRITE REV_ALPHA2 :STREAM MIDOLE)  (WRITE-CHAR #\NEWLINE MIDOLE)))    ; WRITE REVERB ALPHA2 TO FILE MIODLE
;
      (BOUNO_PROMPT_MOD)                                                          ; CALL BOUND_PROMPT_MOO TO PROMPT
;                                                                                 ;   FOR BOUNDARY RETURNS
;
      (TERPRI)
      (PRINC "Is a quantized representation of values (Q) or a representation")   ; PROMPT FOR TYPE OF GRAPH DESIREO
      (PRINC " that shows only the presence of a signal (P) desired? [Q/P] ")
;
      (COND ((EDUAL 'Q (REAO)) (SETQ DUANT 1)))
      (WRITE QUANT :STREAM MIDOLE)   (WRITE-CHAR #\NEWLINE MIOOLE))               ; WRITE DUANT TO FILE MIDDLE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN ASK_SIG ()                                                                  ; FUNCTION TO ASK FOR DESIRED NUMBER
;                                                                                  ;   OF EACH TYPE OF SIGNAL
;
      (SETQ SIGS_PLACE 1)                                                          ; SET FLAG FOR SIGNALS
      (WRITE SIGS_PLACE :STREAM MIDDLE)   (WRITE-CHAR #\NEWLINE MIDDLE)            ; WRITE SIGS_PLACE TO FILE MIDDLE
;
      (TERPRI)                                                                     ; PROMPT FOR DESIRED NUMBER OF EACH
      (PRINC "Enter the number of 'fat' L signals desired (MAX 12) ... ")          ;   TYPE OF SIGNAL
;
                                  (SETQ SET_NUM_FL (READ))
                                  (MIN_MAX_WRITE SET_NUM_FL 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of 'fat' O signals desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_FO (READ))
                                  (MIN_MAX_WRITE SET_NUM_FO 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of 'fat' T signals desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_FT (READ))
                                  (MIN_MAX_WRITE SET_NUM_FT 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of L signals desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_L (READ))
                                  (MIN_MAX_WRITE SET_NUM_L 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of O signals desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_O (READ))
                                  (MIN_MAX_WRITE SET_NUM_O 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of T signals desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_T (READ))
                                  (MIN_MAX_WRITE SET_NUM_T 0 12 1)
      (TERPRI)
      (PRINC "Enter the number of bars desired (MAX 12) ... ")
;
                                  (SETQ SET_NUM_B (READ))
                                  (MIN_MAX_WRITE SET_NUM_B 0 12 1)
      (TERPRI)
      (PRINC "Enter the desired mean for ")
      (PRINC "the level of these signals (MAX 255) ... ")
;
                                  (SETQ SIG_MU (READ))
                                  (MIN_MAX_WRITE SIG_MU 1 255 100)
;
      (TERPRI)
      (PRINC "Are constant level signals (C) or ")
      (PRINC "signal level that vary randomly (R) desired? (C/R)   ")
;
                                  (SETQ RAN_SIG_LEV (READ))
                                  (WRITE RAN_SIG_LEV :STREAM MIDDLE) (WRITE-CHAR #\NEWLINE MIDDLE))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN MIN_MAX_WRITE (SET_NUM_ID MIN_VAL MAX_VAL DEF_VAL)                          ; FUNCTION TO CHECK BOUNDS OF VALUES
;                                                                                  ;   AND WRITE THE VALUES TO FILE MIDDLE
      (COND ((NUMBERP SET_NUM_ID)
             (COND ((< SET_NUM_ID MIN_VAL) (SETQ SET_NUM_ID MIN_VAL)))             ; SET MINIMUM NUMBER OF SIGNALS MIN_VAL
             (COND ((> SET_NUM_ID MAX_VAL) (SETQ SET_NUM_ID MAX_VAL))))            ; SET MAXIMUM NUMBER OF SIGNALS MAX_VAL
            (T  (SETQ SET_NUM_ID DEF_VAL)))                                        ; SET DEFAULT VALUE
;
      (WRITE SET_NUM_ID :STREAM MIDDLE)   (WRITE-CHAR #\NEWLINE MIDDLE))           ; WRITE NUMBER OF SIGNALS TO FILE MIDDLE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(OEFUN ASK_N ()                                                                              ; FUNCTION TO ASK FOR MEAN
;                                                                                            ;   OF NOISE LEVEL
;
     (WRITE FLAT_NOISE :STREAM MIDDLE)   (WRITE-CHAR #\NEWLINE MIDDLE)                      ; WRITE FLAT_NOISE TO FILE MIDDLE
;
      (PRINC "Enter the desired mean for ")                                                  ; PROMPT FOR MEAN OF NOISE
      (PRINC "the level of the noise (MAX 255) ... ")
;
      (SETQ FN_MU (READ))
      (MIN_MAX_WRITE FN_MU 1 255 20))                                                        ; CALL MIN_MAX_WRITE TO CHECK RANGES
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN ASK_REV ()                                                                            ; FUNCTION TO ASK FOR REVERBERATION
;                                                                                            ; PARAMETERS
;
             (WRITE REV_WEIGHT_NOISE :STREAM MIOOLE)   (WRITE-CHAR #\NEWLINE MIOOLE)         ; WRITE REV_WEIGHT_NOISE TO FILE MIDDLE
;
             (PRINC "Enter the desired mean for ")                                           ; PROMT FOR REVERBERATION MEAN
             (PRINC "the level of the reverberation (MAX 255) ... ")
;
                              (SETQ REV_WN_MU (READ))
                              (MIN_MAX_WRITE REV_WN_MU 1 255 200)                            ; CALL MIN_MAX_WRITE TO CHECK RANGES
;
             (TERPRI)
             (PRINC "Enter the desired value to determine ")                                 ; PROMPT FOR REVERBERATION SHAPE
             (PRINC "the shape of the reverberation (MAX 64) ... ")
;
                              (SETQ REV_ALPHA (READ))
                              (MIN_MAX_WRITE REV_ALPHA 1 64 5)                               ; CALL MIN_MAX_WRITE TO CHECK RANGES
;
             (TERPRI)
             (PRINC "Enter the desired mean for ")                                           ; PROMPT FOR NOISE MEAN
             (PRINC "the random component of the reverberation (MAX 255) ... ")
;
                              (SETO REV_ALPHA2 (READ))
                              (MIN_MAX_WRITE REV_ALPHA2 1 255 50))                           ; CALL MIN_MAX_WRITE TO CHECK RANGES
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN SET_SIGS (SET_SIG_FL SET_SIG_FO SET_SIG_FT SET_SIG_L SET_SIG_O SET_SIG_T SET_SIG_H)    ; FUNCTION TO EXECUTE SIGNAL GENERATION
;
        (PICK_SIG_FL SET_SIG_FL)                                                              ; CALL FUNCTION TO PICK THE RANDOM
        (PICK_SIG_FO SET_SIG_FO)                                                              ;   SIGNAL POSITIONS
        (PICK_SIG_FT SET_SIG_FT)
        (PICK_SIG_L SET_SIG_L)
        (PICK_SIG_O SET_SIG_O)
        (PICK_SIG_T SET_SIG_T)
        (PICK_SIG_H SET_SIG_H)
;
        (SETQ LOT_SIGS (APPEND SIG_FL SIG_FO SIG_FT SIG_L SIG_O SIG_T SIG_B ADO_SIGS))       .; PUT THE SIGNALS ALL IN ONE LIST
;
        (DELETE NIL LOT_SIGS)                                                                 ; DELETE ANY NILS FROM SIGNAL LIST
;
        (CHECKEO LOT_SIGS)                                                                    ; CALL FUNCTION TO CHECK FOR
                                                                                              ;   DUPLICATE POINTS IN SIGNAL LIST
;
        (TEST_RANGE_DOPPLER_MAP))                                                             ; CALL FUNCTION TO DISPLAY SIGNALS
;                                                                                             ;   ON MAP
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN CHECKED (CHECK_LIST)                                                   ; FUNCTION TO CHECK FOR DUPLICATE POINTS
;                                                                             ;   IN SIGNAL LIST
   (PROG (TEST)
;
   (SETQ TEST ())
   (SETQ APPEAR ())
;
      LOOP
;
  (COND ((EOUAL CHECK_LIST ()) (RETURN CHECK_LIST))                           ; WHEN ENTIRE LIST IS CHECKED, QUIT
     (T
;
        (SETO PRE_LENGTH (LENGTR CHECK_LIST))                                 ; FIND LENGTH OF CHECK_LIST
        (SETQ TEST (CAR CHECK_LIST))                                          ; LOOK AT FIRST ELEMENT OF CHECK_LIST
        (SETO CHECK_LIST (DELETE TEST CHECK_LIST))                            ; DELETE ALL OCCURENCES OF FIRST_ELEMENT
        (SETQ POST_LENGTH (LENGTH CHECK_LIST))                                ; FIND LENGTH OF SHORTENED CRECK_LIST
;
        (SETQ DIFF (- PRE_LENGTH POST_LENGTH))                                ; DIFFERENCE IN LENGTHS OF TWO LISTS
;
        (SETQ APPEAR (CONS TEST (CONS DIFF APPEAR)))                          ; SAVE NUMBER OF OCCURENCES OF EACH ELEMENT
;
        (GO LOOP)))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_FL (NUM_SIG_FL)                                          ; FUNCTION TO RANDOMLY PLACE 'FAT' L SIGNALS
;
       (SETQ SIG_FL (PICK_HELPER NUM_SIG_FL 'FL 'PLACE_SIG_FL)))          ; CALL PICK_RELPER FUNCTION TO PLACE SIGNALS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_FL (CENTER)                                             ; FUNCTION TO PLACE DESIGN OF 'FAT' L
;
               (CONS (NTH (- CENTER 128) IN_LINE_FT)                     ;    **
               (CONS (NTH (- CENTER 127) IN_LINE_FT)                     ; ******
               (CONS (NTH (- CENTER 126) IN_LINE_FT)                     ; **##**
               (CONS (NTH (- CENTER 125) IN_LINE_FT)                     ; ******
               (CONS (NTH (- CENTER 124) IN_LINE_FT)                     ;   ************
               (CONS (NTH (- CENTER 123) IN_LINE_FT)
;                                                                        ;   ## = CENTER
               (CONS (NTH (- CENTER 65) IN_LINE_FT)
               (CONS (NTH (- CENTER 64) IN_LINE_FT)
               (CONS (NTH (- CENTER 63) IN_LINE_FT)
;
               (CONS (NTH (- CENTER 1) IN_LINE_FT)
               (CONS (NTH CENTER IN_LINE_FT)
               (CONS (NTH (+ CENTER 1) IN_LINE_FT)
;
               (CONS (NTH (+ CENTER 63) IN_LINE_FT)
               (CONS (NTH (+ CENTER 64) IN_LINE_FT)
               (CONS (NTH (+ CENTER 65) IN_LINE_FT)
;
               (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_HELPER (NUM_SIG_ID ID PLACE_SIG_ID)                              ; FUNCTION TO RANDOMLY PLACE
;                                                                            ;   SIGNALS ON A RANGE DOPPLER MAP
     (PROG (COUNTR)
;
          (SETQ COUNTR NUM_SIG_ID)
          (SETQ SIG_CEN ())
          (SETQ SIG_ID ())
;
           LOOP
;
              (COND ((EQUAL COUNTR 0) (RETURN SIG_ID))                       ; WHEN ALL SIGNALS PLACED, QUIT
;
                    (T
                     (SETQ SIG_CEN ())                                       ; RESET SIG_CEN
;
                     (COND ((EQUAL ID 'H)                                    ; PICK RANDOM CENTER VALUE
                            (SETQ SIG_CENTER (RANDOM 39)))                   ;   WITHIN BOUNDS OF MAP
                           (T
                            (SETQ SIG_CENTER (WITHIN_BOUNDS (RANDOM 2560)))))
;
                     (SETQ SIG_CEN (CONS ID (CONS SIG_CENTER SIG_CEN)))      ; LABEL TYPE OF SIGNAL WITH CENTER VALUE
;
                     (SETQ CEN_SIG (CONS SIG_CEN CEN_SIG))                   ; ADD LABEL TO CEN_SIG LIST
;
                     (SETQ SIG_ID (APPEND SIG_ID (FUNCALL PLACE_SIG_ID SIG_CENTER)))  ; CALL PLACE_SIG_* FUNCTION TO
                                                                             ;   GENERATE SIGNAL AND ADD TO SIG_ID
;
                     (SETQ COUNTR (- COUNTR 1))))                            ; DECREASE COUNTER
;
          (GO LOOP)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN WITHIN_BOUNDS (RAND)                                                  ; FUNCTION TO CHECK COLUMN POSITION OF
;                                                                            ;   SIGNAL CENTERS TO ENSURE NO SIGNAL
       (PROG ()                                                              ;   OVERLAPS OVER 'EDGE' OF MAP
;
             LOOP
;
                (COND ((MEMBER (- RAND  (* 64 (TRUNCATE (/ RAND 64))))
                                '(0 1 2 3 4 5 6 7 58 59 60 61 62 63))        ; IF COLUMN POSITION = -6 TO 7 (MOD 64)
                       (SETQ RAND (RANDOM 2560))                             ;    PLACE SIGNAL CENTER IN A NEW COLUMN
                       (GO LOOP))
                      (T (RETURN RAND)))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_FO (NUM_SIG_FO)                                              ; FUNCTION TO RANDOMLY PLACE 'FAT' O SIGNALS
;
      (SETQ SIG_FO (PICK_HELPER NUM_SIG_FO 'FO 'PLACE_SIG_FO)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_FO (CENTER)                                                 ; FUNCTION TO PLACE DESIGN OF 'FAT' O
;
             (CONS (NTH (- CENTER 128) IN_LINE_FT)                           ;    **
;                                                                            ; ******
             (CONS (NTH (- CENTER 65) IN_LINE_FT)                            ; **##**
             (CONS (NTH (- CENTER 64) IN_LINE_FT)                            ; ******
             (CONS (NTH (- CENTER 63) IN_LINE_FT)                            ;   **
;
             (CONS (NTH (- CENTER 1) IN_LINE_FT)                             ;   ## = CENTER
             (CONS (NTH CENTER IN_LINE_FT)
             (CONS (NTH (+ CENTER 1) IN_LINE_FT)
;
             (CONS (NTH (+ CENTER 63) IN_LINE_FT)
             (CONS (NTH (+ CENTER 64) IN_LINE_FT)
             (CONS (NTH (+ CENTER 65) IN_LINE_FT)
;
             (CONS (NTH (+ CENTER 128) IN_LINE_FT) ())))))))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_FT (NUM_SIG_FT)                                              ; FUNCTION TO RANDOMLY PLACE 'FAT' T SIGNALS
;
      (SETQ SIG_FT (PICK_HELPER NUM_SIG_FT 'FT 'PLACE_SIG_FT)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_FT (CENTER)                                                 ; FUNCTION TO PLACE DESIGN OF 'FAT' T
;
             (CONS (NTH (- CENTER 128) IN_LINE_FT)                           ; ******************
;                                                                            ;       ******
             (CONS (NTH (- CENTER 65) IN_LINE_FT)                            ;       **##**
             (CONS (NTH (- CENTER 64) IN_LINE_FT)                            ;       ******
             (CONS (NTH (- CENTER 63) IN_LINE_FT)                            ;         **
;
             (CONS (NTH (- CENTER 1) IN_LINE_FT)                             ;         ## = CENTER
             (CONS (NTH CENTER IN_LINE_FT)
             (CONS (NTH (+ CENTER 1) IN_LINE_FT)
;
             (CONS (NTH (+ CENTER 63) IN_LINE_FT)
             (CONS (NTH (+ CENTER 64) IN_LINE_FT)
             (CONS (NTH (+ CENTER 65) IN_LINE_FT)
;
             (CONS (NTH (+ CENTER 124) IN_LINE_FT)
             (CONS (NTH (+ CENTER 125) IN_LINE_FT)
             (CONS (NTH (+ CENTER 126) IN_LINE_FT)
             (CONS (NTH (+ CENTER 127) IN_LINE_FT)
             (CONS (NTH (+ CENTER 128) IN_LINE_FT)
             (CONS (NTH (+ CENTER 129) IN_LINE_FT)
             (CONS (NTH (+ CENTER 130) IN_LINE_FT)
             (CONS (NTH (+ CENTER 131) IN_LINE_FT)
             (CONS (NTH (+ CENTER 132) IN_LINE_FT) ()))))))))))))))))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_L (NUM_SIG_L)                                                      ; FUNCTION TO RANDOMLY PLACE L SIGNALS
;
       (SETQ SIG_L (PICK_HELPER NUM_SIG_L 'L 'PLACE_SIG_L)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_L (CENTER)                                                        ; FUNCTION TO PLACE DESIGN OF L
;
               (CONS (NTH (- CENTER 128) IN_LINE_FT)                               ; **
               (CONS (NTH (- CENTER 127) IN_LINE_FT)                               ; **
               (CONS (NTH (- CENTER 126) IN_LINE_FT)                               ; ##
               (CONS (NTH (- CENTER 125) IN_LINE_FT)                               ; **
;                                                                                  ; ********
               (CONS (NTH (- CENTER 64) IN_LINE_FT)
;                                                                                  ;   ## = CENTER
               (CONS (NTH CENTER IN_LINE_FT)
;
               (CONS (NTH (+ CENTER 64) IN_LINE_FT)
;
               (CONS (NTH (+ CENTER 128) IN_LINE_FT) ()))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_O (NUM_SIG_O)                                                      ; FUNCTION TO RANDOMLY PLACE O SIGNALS
;
       (SETQ SIG_O (PICK_HELPER NUM_SIG_O 'O 'PLACE_SIG_O)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_O (CENTER)                                                        ; FUNCTION TO PLACE DESIGN OF O
;
               (CONS (NTH (- CENTER 128) IN_LINE_FT)                               ;     **
;                                                                                  ;   **  **
               (CONS (NTH (- CENTER 65) IN_LINE_FT)                                ;   **##**
               (CONS (NTH (- CENTER 63) IN_LINE_FT)                                ;   **  **
;                                                                                  ;     **
               (CONS (NTH (- CENTER 1) IN_LINE_FT)
               (CONS (NTH (+ CENTER 1) IN_LINE_FT)                                 ;     ## = CENTER
;
               (CONS (NTH (+ CENTER 63) IN_LINE_FT)
               (CONS (NTH (+ CENTER 65) IN_LINE_FT)
;
               (CONS (NTH (+ CENTER 128) IN_LINE_FT) ()))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_T (NUM_SIG_T)                                        ; FUNCTION TO RANDOMLY PLACE T SIGNALS
;
       (SETQ SIG_T (PICK_HELPER NUM_SIG_T 'T 'PLACE_SIG_T)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_T (CENTER)                                          ; FUNCTION TO PLACE DESIGN OF T
;
               (CONS (NTH (- CENTER 128) IN_LINE_FT)                 ; **************
;                                                                    ;        **
               (CONS (NTH (- CENTER 64) IN_LINE_FT)                  ;        ##
;                                                                    ;        **
               (CONS (NTH CENTER IN_LINE_FT)                         ;        **
;
               (CONS (NTH (+ CENTER 64) IN_LINE_FT)                  ;        ## = CENTER
;
               (CONS (NTH (+ CENTER 125) IN_LINE_FT)
               (CONS (NTH (+ CENTER 126) IN_LINE_FT)
               (CONS (NTH (+ CENTER 127) IN_LINE_FT)
               (CONS (NTH (+ CENTER 128) IN_LINE_FT)
               (CONS (NTH (+ CENTER 129) IN_LINE_FT)
               (CONS (NTH (+ CENTER 130) IN_LINE_FT)
               (CONS (NTH (+ CENTER 131) IN_LINE_FT) ()))))))))))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PICK_SIG_B (NUM_SIG_B)                                        ; FUNCTION TO RANDOMLY PLACE BARS
;
       (SETQ SIG_B (PICK_HELPER NUM_SIG_B 'B 'PLACE_SIG_B)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN PLACE_SIG_B (CENTER)                                          ; FUNCTION TO PLACE DESIGN OF BAR
;
          (PROG (RESULT TIM)                                         ; ******************************************...
;                                                                    ; ******************************************...
               (SETQ TIM 0 )                                         ; ##****************************************...
               (SETQ RESULT ())                                      ; ******************************************...
               (SETQ CENTER (* CENTER 64))                           ; ******************************************...
;
           LOOP                                                      ; ## = CENTER
;
                (COND ((> TIM 63) (RETURN RESULT))
;
                  (T
;
                 (SETQ RESULT (CONS (NTH (+ (- CENTER 128) TIM) IN_LINE_FT)
                              (CONS (NTH (+ (- CENTER 64) TIM) IN_LINE_FT)
                              (CONS (NTH (+ CENTER TIM) IN_LINE_FT)
                              (CONS (NTH (+ CENTER (+ 64 TIM)) IN_LINE_FT)
                              (CONS (NTH (+ CENTER (+ 128 TIM )) IN_LINE_FT) RESULT))))))
;
                   (SETQ TIM (+ TIM 1))))
;
                 (GO LOOP)))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN TEST_RANGE_DOPPLER_MAP ()                                            ; FUNCTION TO CALL CELL_LEVEL FUNCTION
;
   (PROG (COUNTER)
      (SETQ COUNTER 0)                                                       ; INITIALIZE COUNTER
      (SETQ NUMB 2497)                                                       ; START AT FIRST ELEMENT IN ROW 40
      (SETQ COL_NUMB 1)                                                      ; SET COLUMN NUMBER TO 1
;
      LOOP                                                                   ; LOOP TO CALL CELL_LEVEL FUNCTION
;                                                                            ;     FROM ROW 40 TO ROW 1
;
            (SETQ COUNTER (+ COUNTER 1))                                     ; INCREMENT COUNTER FOR COLUMN POSITION
;
            (COND   ((< COUNTER 65)                                          ; FOR ALL 64 COLUMN POSITIONS IN ONE ROW,
                        (CELL_LEVEL NUMB)                                    ;   CALL CELL_LEVEL
                        (SETQ NUMB (+ NUMB 1))                               ;   INCREMENT NUMBER
                        (SETQ COL_NUMB (+ COL_NUMB 1))                       ;   INCREMENT COLUMN NUMBER
                   (GO LOOP))                                                ;   REPEAT LOOP
;
            (T
                   (SETQ NUMB (- NUMB 128))                                  ;   GO TO BEGINNING OF PREVIOUS ROW
                   (SETQ COL_NUMB 1)                                         ;   RESET COLUMN NUMBER
                   (COND ((< NUMB 1) (RETURN NUMB))                          ;   IF ALL ROWS CALLED, QUIT
;
                    (T                                                       ; IF MORE ROWS TO CALL,
                         (SETQ COUNTER 0)                                    ;    RESET COUNTER FOR COLUMN POSITION
                         (GO LOOP)))))))                                     ;    REPEAT LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN CELL_LEVEL (POSIT)                                                    ; FUNCTION TO ASSIGN VALUES OF SIGNAL, NOISE
;                                                                            ;   AND REVERBERATION TO A SINGLE CELL
;
    (SETQ POSIT (- POSIT 1))                                                 ; DECREMENT POSIT TO ACCOUNT FOR STARTING POSITION OF 0
    (SETQ SUM_VALUE 0)                                                       ; INITIALIZE SUM VALUE
    (SETQ TOT_SIG_VAL 0)                                                     ; INITIALIZE TOTAL SIGNAL VALUE
    (SETQ REV_W_NOISE_VALUE 0)                                               ; INITIALZE WEIGHTED NOISE VALUE
    (SETQ F_NOISE_VALUE 0)                                                   ; INITIALIZE FLAT NOISE VALUE
    (SETQ BD_VAL_SQ 0)                                                       ; INITIALIZE BOUNDARY VALUES
;
    (IF_DESIRED POSIT)                                                       ; PLACE SIGNALS IF DESIRED
;
    (COND ((EQUAL REV_WEIGHT_NOISE 1)                                        ; IF WEIGHTED NOISE IS TO APPEAR,
           (DISTRIBUTE_REV_WN REV_WN_MU)                                     ;    CALL FOR WEIGHTED DISTRIBUTION
           (SETQ REV_W_NOISE_VALUE (* REV_W_NOISE_VALUE REV_W_NOISE_VALUE))));    SQUARE THE VALUE OF REV_W_NOISE_VALUE
;
    (COND ((EQUAL FLAT_NOISE 1)                                              ; IF FLAT NOISE IS TO APPEAR,
           (DISTRIBUTE_FN_MU)                                                ;    CALL FOR RANDOM DISTRIBUTION
           (SETQ F_NOISE_VALUE (* VALUE VALUE))))                            ;    SQUARE THE VALUE OF F_NOISE_VALUE
;
    (COND ((EQUAL BOUND_FLAG 1)                                              ; IF BOUNDARY VALUES DESIRED,
           (CALC_BD_VAL POSIT BOUND_RET                                      ;    CALCULATE BOUNDARY VALUES
                       BOUND_GEN BOUND_CONS_LEV BOUND_AVG_LEV)))
;
    (SETQ SUM_VALUE (+ (+ (+ TOT_SIG_VAL REV_W_NOISE_VALUE) F_NOISE_VALUE)     ; ADD VALUE OF SIGNALS, REVERBERATION, NOISE,
                     BD_VAL_SQ))                                             ;    AND BOUNDARY RETURN
    (SETQ FINAL_VALUE (TRUNCATE (SQRT SUM_VALUE)))                           ;   TAKE SQUARE ROOT OF SUM OF SQUARES
    (SETQ OUTPUT_FILE (CONS FINAL_VALUE OUTPUT_FILE))                        ;   ADD ELEMENT TO OUTPUT_FILE
    (SETQ OUTPUT_FILE1 (CONS FINAL_VALUE OUTPUT_FILE1))                      ;   ADD ELEMENT TO OUTPUT_FILE1
    (COND ((> FINAL_VALUE 256) (SETQ FINAL_VALUE 256)))                      ;   MAXIMUM FINAL_VALUE IS 256 FOR COMPUTATION OF
                                                                             ;     OF QUANTIZED VALUES
;
    (COND ((EQUAL QUANT 1)
           (SETQ STARS (+ (TRUNCATE (/ FINAL_VALUE 32)) 1))                  ; QUANTIZE THE VALUE = (VALUE / 32) + 1
           (SETQ OUTPUT_LIST (CONS STARS OUTPUT_LIST))))                     ; ADD QUANTIZED VALUE TO OUTPUT_LIST
;
    (SETQ COUNT (+ COUNT 1))  (SETQ KOUNT (+ KOUNT 1))                       ; INCREMENT COUNTERS FOR OUTPUT FILES
;
    (COND ((EQUAL COUNT 64)                                                  ; WHEN COUNTER IS AT 64,
           (SETQ OUTPUT_LIST (REVERSE OUTPUT_LIST))                          ;   REVERSE OUTPUT_LIST
           (WRITE OUTPUT_LIST :STREAM LOOK)  (WRITE-CHAR #\NEWLINE LOOK)     ;   WRITE LINE TO FILE LOOK AND SKIP TO NEXT LINE
           (SETQ OUTPUT_FILE1 (REVERSE OUTPUT_FILE1))                        ;   REVERSE OUTPUT_FILE1
           (WRITE OUTPUT_FILE1 :STREAM PLOT3D) (WRITE-CHAR #\NEWLINE PLOT3D) ;   WRITE LINE TO FILE PLOT3D AND SKIP TO NEXT LINE
           (SETQ OUTPUT_FILE1 ())                                            ;   RESET OUTPUT_FILE1
           (SETQ COUNT 0)                                                    ;   RESET COUNTER TO 0
           (PRINC "      ")                                                  ;   SKIP TO NEXT LINE ON SCREEN
           (SETQ OUTPUT_LIST ())))                                           ;   RESET OUTPUT_LIST
;
    (COND ((EQUAL KOUNT 32)                                                  ; WHEN COUNTER IS AT 32,
           (SETQ OUTPUT_FILE (REVERSE OUTPUT_FILE))                          ;   REVERSE OUTPUT_FILE
           (WRITE OUTPUT_FILE :STREAM REVIDEN) (WRITE-CHAR #\NEWLINE REVIDEN);   WRITE LINE TO FILE REVIDEN AND SKIP TO NEXT LINE
           (SETQ OUTPUT_FILE ())                                             ;   RESET OUTPUT_FILE
           (SETQ KOUNT 0))))                                                 ;   RESET COUNTER TO 0
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN IF_DESIRED (RD_POS)                                          ; FUNCTION TO PLACE SIGNALS ON MAP
;
   (COND ((EQUAL SIGS_PLACE 1)                                      ; PLACE SIGNALS IF DESIRED
;
      (COND ((EQUAL (NTH RD_POS IN_LINE_FT)                         ; CHECK IF ELEMENT IS IN SIGNAL 'POINT' LIST
                  (CAR (MEMBER (NTH RD_POS IN_LINE_FT) LOT_SIGS)))
             (SETQ TIME_APP (CADR (MEMBER (NTH RD_POS IN_LINE_FT) APPEAR))) ;   READ HOW MANY TIMES IT APPEARS
;
             (ITERATE)                                              ;   CALL ITERATE FUNCTION
;
             (COND ((EQUAL QUANT 0)
                    (SETQ OUTPUT_LIST (CONS YES_STARS OUTPUT_LIST))))   ; ADD ELEMENT TO OUTPUT_LIST
             (PRINC "**"))                                          ; PRINTS ** TO SCREEN IN POSITION OF SIGNAL
;
      (T
             (PRINC "  ")                                           ; PRINTS BLANK TO SCREEN WHERE NO SIGNAL EXISTS
             (COND ((EQUAL QUANT 0)
                    (SETQ OUTPUT_LIST (CONS NO_STARS OUTPUT_LIST))))))))) ; ADD ELEMENT TO OUTPUT_LIST
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN ITERATE ()                                                   ; FUNCTION TO RECALL DISTRIBUTE FUNCTION
;                                                                   ;   FOR OVERLAPPING SIGNALS
  (PROG (TIMES)
;
         (SETQ TIMES TIME_APP)                                      ; SET TIMES TO TIME_APP
;
         LOOP
;
            (COND ((EQUAL TIMES 0) (RETURN TIMES))                  ; IF POINT APPEARS 0 ADDITIONAL TIMES, QUIT
                  (T                                                ;   OTHERWISE,
;
                   (COND ((EQUAL RAN_SIG_LEV 'C)                    ; IF SIGNAL IS TO BE CONSTANT LEVEL,
                          (SETQ VALUE SIG_MU))                      ;   SET SIGNAL TO SIGNAL MEAN
                         (T
                          (DISTRIBUTE SIG_MU)))                     ; CALL FOR RANDOM DISTRIBUTION OF SIGNAL LEVELS
;
                   (SETQ TOT_SIG_VAL (+ (* VALUE VALUE) TOT_SIG_VAL)) ;   ADD VALUE OF SIGNAL SQUARED
                   (SETQ TIMES (- TIMES 1))                         ;   DECREASE TIMES
         (GO LOOP)))))                                              ; REPEAT LOOP
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN DISTRIBUTE (MU)                                              ; FUNCTION TO FIND A RAYLEIGH DISTRIBUTION
;
      (SETQ X (* 2 (* MU MU)))                                      ; X = 2 * MU * MU
      (SETQ Y (LOG (+ (RANDOM 1.0) 0.00001)))                       ; Y = LN ((RANDOM 1) + 0.00001)
      (SETQ Z (* -1 (* X Y)))                                       ; Z = - (X * Y)
      (SETQ VALUE (TRUNCATE (SQRT Z))))                             ; VALUE = Z ** 1/2
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN DISTRIBUTE_REV_WN (REV_WN_MU)                                ; FUNCTION TO DISTRIBUTE REVERB WEIGHTED NOISE
;
      (SETQ FBIN 32)                                                ; INITIALIZE FBIN
;
      (SETQ A (- (+ FBIN REV_ALPHA) COL_NUMB))                      ; A = FBIN + REV_ALPHA - COL_NUMB
      (SETQ H (* (* REV_WN_MU REV_ALPHA) (SQRT (EXP 1))))           ; H = REV_WN_MU * REV_ALPHA * (SQRT E)
      (SETQ C (/ A (* REV_ALPHA REV_ALPHA)))                        ; C = A / (REV_ALPHA * REV_ALPHA)
      (SETQ D (* (/ (* A A) (* 2 (* REV_ALPHA REV_ALPHA))) -1))     ; D = -1 * [(A * A) / (2 * REV_ALPHA * REV_ALPHA)]
      (SETQ E (EXP D))                                              ; E = E ** D
      (SETQ F (* H (* C E)))                                        ; F = H * C * E
;
      (COND ((OR (< A 0) (< F (/ REV_WN_MU 100))) (SETQ REV_W_NOISE_VALUE 0)); NO WEIGHTED NOISE TYPE REVERBERATION VALUE FOR
;                                                                   ; VALUES OF COL_NUMB GREATER THAN (FBIN + REV_ALPHA)
            (T
             (DISTRIBUTE REV_ALPHA2)                                ; CALL DISTRIBUTE FOR RANDOM BACKGROUND NOISE
             (SETQ RAN_NOISE_VALUE VALUE)                           ; SET RAN_NOISE_VALUE TO VALUE GENERATED
             (SETQ REV_W_NOISE_VALUE (+ RAN_NOISE_VALUE F)))))      ; ADD RAN_NOISE_VALUE TO REVERB WEIGHTED NOISE VALUE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN BOUND_PROMPT_MOD ()                                                   ; FUNCTION TO PROMPT FOR BOUNDARY CONDTIONS
;
      (TERPRI)
      (PRINC "Are boundary returns desired? [Y/N) ..................... ")
      (SETQ BOUND_GEN (READ))
      (COND ((EQUAL BOUND_GEN 'Y) (BOUNDARY)                                 ; IF DESIRED, GENERATE BOUNDARY RETURNS
                                  (SETQ BOUND_FLAG 1))
            (T                                                               ; OTHERWISE, QUIT
              (SETQ BOUND_GEN 'NO) (SETQ BOUND_FLAG 0) (SETQ BOUND_RET ()))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN BOUNDARY ()                                                           ; FUNCTION TO PROMPT FOR BOUNDARY CONDITION VALUES
;
            (TERPRI) (PRINC "Platform depth (m) ..................")
            (PRINC "................... ") (SETQ PLATD (READ))
;
            (TERPRI) (PRINC "Platform velocity (m/sec) ...........")
            (PRINC "................... ") (SETQ PLATV (READ))
;
            (TERPRI)
            (PRINC "Water depth (m) ..............................")
            (PRINC "......... ") (SETQ WATERD (READ))
;
            (TERPRI)
            (PRINC "Average level of all boundary returns ........")
            (PRINC "......... ") (SETQ BOUND_AVG_LEV (READ))
;
            (TERPRI)
            (PRINC "Are constant boundary returns desired? [Y/N) ..")
            (PRINC "......... ") (SETQ BOUND_CONS_LEV (READ))
;
            (SURF_GEN PLATD PLATV)                                           ; GENERATE BOUNDARY RETURNS
            (BOTT_GEN PLATD PLATV WATERD)
            (S1B1_GEN PLATD PLATV WATERD)
            (B1S1_GEN PLATD PLATV WATERD)
            (S2B1_GEN PLATD PLATV WATERD)
            (B2S1_GEN PLATD PLATV WATERD)
            (S2B2_GEN PLATD PLATV WATERD)
            (B2S2_GEN PLATD PLATV WATERD)
;
            (SETQ BOUND_RET (APPEND SURF_RET BOTT_RET S1B1_RET B1S1_RET      ; COMBINE ALL COMPONENTS OF BOUNDARY RETURNS
                                    S2B1_RET B2S1_RET S2B2_RET B2S2_RET)))   ;   TO GET BOUND_RET VALUE
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN SURF_GEN (DEPTH1 VELOC)                                               ; FUNCTION TO PROMPT FOR AND COMPUTE SURFACE RETURNS
;
      (SETQ SURF_RET ())
      (TERPRI)
      (PRINC "Are surface boundary returns desired? [Y/N) ............ ")
      (SETQ SURF_RIDGE (READ))
      (COND ((EQUAL SURF_RIDGE 'Y)                                           ; IF DESIRED, GENERATE SURFACE BOUNDARY RETURNS
;
            (PROG (BCOUNT)
                  (SETQ BCOUNT 31)
;
                  LOOP
;
                  (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                             (RETURN SURF_RET))
                        (T
                          (SETQ DELF (* (- 32 BCOUNT) 20))
                          (SETQ ANG (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                          (SETQ STIME (/ DEPTH1 (* 750 (SIN ANG))))
                          (SETQ SURF_RET (APPEND (TRANS_SIG STIME BCOUNT)
                                                  SURF_RET))
                          (SETQ BCOUNT (- BCOUNT 1))
                          (GO LOOP)))))
;
           (T
             (SETQ SURF_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN TRANS_SIG (SIG_CENT BCNT)                                             ; FUNCTION TO REPRESENT TRANSMITTED SIGNAL
;                                                                            ;   IN BOUNDARY RETURN
       (SETQ TRANS_CELLS ())
       (SETQ STINC (* 64 (TRUNCATE (/ SIG_CENT 0.05))))
       (SETQ BCELL (+ (+ BCNT 1) STINC))
       (SETQ BCELLP (+ BCELL 64))
       (SETQ BCELLM 0)
       (SETQ TRANS_CELLS (CONS BCELLP (CONS BCELLM
                                           (CONS BCELL TRANS_CELLS))))
       (DELETE 0 TRANS_CELLS))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN BOTT_GEN (DEPTH1 VELOC DEPTH2)                                    ; FUNCTION TO PROMPT FOR AND COMPUTE BOTTOM RETURNS
;
      (SETQ BOTT_RET ())
      (TERPRI)
      (PRINC "Are bottom boundary returns desired? [Y/N) ............ ")
      (SETQ BOTT_RIDGE (READ))
      (COND ((EQUAL BOTT_RIDGE 'Y)                                       ; IF DESIRED, GENERATE BOTTOM BOUNDARY RETURNS
;
             (PROG (BCOUNT)
                   (SETQ BCOUNT 31)
;
                   LOOP
;
                   (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                              (RETURN BOTT_RET))
                         (T
                           (SETQ DELF (* (- 32 BCOUNT) 20))
                           (SETQ ANG (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                           (SETQ STIME (/ DEPTH1 (* 750 (SIN ANG))))
                           (SETQ BTIME (- (/ DEPTH2 (* 750 (SIN ANG))) STIME))
                           (SETQ BOTT_RET (APPEND (TRANS_SIG BTIME BCOUNT)
                                                  BOTT_RET))
                           (SETQ BCOUNT (- BCOUNT 1))
                           (GO LOOP)))))
;
            (T
              (SETQ BOTT_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN S1B1_GEN (DEPTH1 VELOC DEPTH2)                                    ; FUNCTION TO PROMPT FOR AND COMPUTE
;                                                                        ;     ONE SURFACE AND ONE BOTTOM RETURN
      (SETQ S1B1_RET ())
      (TERPRI)
      (PRINC "Are surf(1)bott(1) boundary returns desired? [Y/N] ..... ")
      (SETQ S1B1_RIDGE (READ))
      (COND ((EQUAL S1B1_RIDGE 'Y)                                       ; IF DESIRED, GENERATE S1B1 BOUNDARY RETURNS
;
             (PROG (BCOUNT)
                   (SETQ BCOUNT 31)
;
                   LOOP
;
                   (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                              (RETURN S1B1_RET))
                         (T
;
                   (SETQ DELF (* (- 32 BCOUNT) 20))
                   (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                   (SETQ SUMD (+ DEPTH1 DEPTH2))
                   (SETQ DIFD (- DEPTH2 DEPTH1))
                   (SETQ ANG2 (ATAN (/ (* SUMD (TAN ANG1)) DIFD)))
                   (SETQ TTN (+ (* DIFD (SIN ANG2)) (* SUMD (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ S1B1_RET (APPEND (TRANS_SIG SB_TIME BCOUNT)
                                          S1B1_RET))
                   (SETQ BCOUNT (- BCOUNT 1))
                   (GO LOOP)))))
;
            (T
              (SETQ S1B1_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN B1S1_GEN (DEPTB1 VELOC DEPTB2)                                                       ; FUNCTION TO PROMPT FOR AND COMPUTE
;                                                                                           ;   ONE BOTTOM AND ONE SURFACE RETURN
      (SETQ B1S1_RET ())
      (TERPRI)
      (PRINC "Are bott(1)surf(1) boundary returns desired? [Y/N] ..... ")
      (SETQ B1S1_RIDGE (READ))
      (COND ((EQUAL B1S1_RIDGE 'Y)                                                          ; IF DESIRED, GENERATE B1S1 BOUNDARY RETURNS
;
             (PROG (BCOUNT)
                   (SETQ BCOUNT 31)
;
                   LOOP
;
                   (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                              (RETDRN B1S1_RET))
                         (T
;
                   (SETQ DELF (* (- 32 BCOUNT) 20))
                   (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                   (SETQ DIF2 (- (* 2 DEPTB2) DEPTB1))
                   (SETQ ANG2 (ATAN (/ (* DIF2 (TAN ANG1)) DEPTB1)))
                   (SETQ TTN (+ (* DEPTB1 (SIN ANG2)) (* DIF2 (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ B1S1_RET (APPEND (TRANS_SIG SB_TIME BCOUNT)
                                          B1S1_RET))
                   (SETQ BCOUNT (- BCOUNT 1))
                   (GO LOOP)))))
;
            (T
              (SETQ B1S1_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN S2B1_GEN (DEPTB1 VELOC DEPTB2)                                                       ; FUNCTION TO PROMPT FOR AND COMPUTE
;                                                                                           ;   TWO SURFACE AND ONE BOTTOM RETURN
      (SETQ S2B1_RET ())
      (SETQ S2B1_RET1 ())
      (SETQ S2B1_RET2 ())
      (TERPRI)
      (PRINC "Are surf(2)bott(1) boundary returns desired? [Y/N] ..... ")
      (SETQ S2B1_RIDGE (READ))
      (COND ((EQUAL S2B1_RIDGE 'Y)                                                          ; IF DESIRED, GENERATE S2B1 BOUNDARY RETURNS
;
             (PROG (BCOUNT)
                   (SETQ BCOUNT 31)
;
                   LOOP
;
                   (COND ((OR (EQUAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                              (SETQ S2B1_RET (APPEND S2B1_RET1 S2B1_RET2))
                              (RETURN S2B1_RET))
                         (T
;
                   (SETQ DELF (* (- 32 BCOUNT) 20))
                   (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                   (SETQ SUM2 (+ (* 2 DEPTB2) DEPTB1))
                   (SETQ ANG2 (ATAN (/ (* SUM2 (TAN ANG1)) DEPTB1)))
                   (SETQ TTN (+ (* DEPTB1 (SIN ANG2)) (* SUM2 (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ S2B1_RET1 (APPEND (TRANS_SIG SB_TIME BCOUNT)
                                           S2B1_RET1))
;
                   (SETQ TTN2 (* 2 (+ DEPTB1 DEPTB2)))
                   (SETQ TTD2 (* 1500 (SIN ANG1)))
                   (SETQ SB2_TIME (/ TTN2 TTD2))
                   (SETQ S2B1_RET2 (APPEND (TRANS_SIG SB2_TIME BCOUNT)
                                           S2B1_RET2))
;
                   (SETQ BCOUNT (- BCOUNT 1))
                   (GO LOOP)))))
;
            (T
              (SETQ S2B1_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN B2S1_GEN (DEPTB1 VELOC DEPTB2)                                          ; FUNCTION TD PRDMPT FDR AND COMPUTE
;                                                                              ;    TWD BDTTDM AND DNE SURFACE RETURN
      (SETQ B2S1_RET ())
      (SETQ B2S1_RET1 ())
      (SETQ B2S1_RET2 ())
      (TERPRI)
      (PRINC "Are bott(2)surf(1) boundary returns desired? [Y/N] ..... ")
      (SETQ B2S1_RIDGE (READ))
      (CDND ((EQUAL B2S1_RIDGE 'Y)                                             ; IF DESIRED, GENERATE B2S1 BOUNDARY RETURNS
;
             (PROG (BCDUNT)
                   (SETQ BCDUNT 31)
;
                   LOOP
;
                   (CDND ((DR (EQUAL BCDUNT 0) (<= BCDUNT (- 32 (/ VELOC 2))))
                               (SETQ B2S1_RET (APPEND B2S1_RET1 B2S1_RET2))
                               (RETURN B2S1_RET))
                         (T
;
                   (SETQ DELF (* (- 32 BCDUNT) 20))
                   (SETQ ANG1 (ACDS (- 1 (/ DELF (* 10 VELOC)))))
                   (SETQ DIF3 (- (* 3 DEPTH2) DEPTH1))
                   (SETQ DIFD (- DEPTB2 DEPTH1))
                   (SETQ ANG2 (ATAN (/ (* DIF3 (TAN ANG1)) DIFD)))
                   (SETQ TTN (+ (* DIFD (SIN ANG2)) (* DIF3 (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ B2S1_RET1 (APPEND (TRANS_SIG SB_TIME BCDUNT)
                                           B2S1_RET1))
;
                   (SETQ TTN2 (* 2 (- (* 2 DEPTB2) DEPTB1)))
                   (SETQ TTD2 (* 1500 (SIN ANG1)))
                   (SETQ SB2_TIME (/ TTN2 TTD2))
                   (SETQ B2S1_RET2 (APPEND (TRANS_SIG SB2_TIME BCDUNT)
                                           B2S1_RET2))
;
                   (SETQ BCDUNT (- BCDUNT 1))
                   (GD LODP)))))
;
            (T
              (SETQ B2S1_RIDGE 'ND))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN S2B2_GEN (DEPTB1 VELOC DEPTH2)                                          ; FUNCTIDN TD PRDMPT FDR AND CDMPUTE
;                                                                              ;    TWD SURFACE AND TWQ BQTTQM RETURNS
      (SETQ S2B2_RET ())
      (SETQ S2B2_RET1 ())
      (SETQ S2B2_RET2 ())
      (TERPRI)
      (PRINC "Are surf(2)bott(2) boundary returns desired? [Y/N] ..... ")
      (SETQ S2B2_RIDGE (READ))
      (CDND ((EQUAL S2B2_RIDGE 'Y)                                             ; IF DESIRED, GENERATE S2B2 BOUNDARY RETURNS
;
             (PRDG (BCDUNT)
                   (SETQ BCDUNT 31)
;
                   LOOP
;
                   (CDND ((DR (EQUAL BCDUNT 0) (<= BCDUNT (- 32 (/ VELOC 2))))
                               (SETQ S2B2_RET (APPEND S2B2_RET1 S2B2_RET2))
                               (RETURN S2B2_RET))
                         (T
;
                   (SETQ DELF (* (- 32 BCQUNT) 20))
                   (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELDC)))))
                   (SETQ SUM3 (+ (* 3 DEPTB2) DEPTB1))
                   (SETQ DIFD (- DEPTB2 DEPTB1))
                   (SETQ ANG2 (ATAN (/ (* SUM3 (TAN ANG1)) DIFD)))
                   (SETQ TTN (+ (* DIFD (SIN ANG2)) (* SUM3 (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ S2B2_RET1 (APPEND (TRANS_SIG SB_TIME BCDUNT)
                                           S2B2_RET1))
;
                   (SETQ SUMD2 (+ DEPTB1 DEPTH2))
                   (SETQ ANG22 (ATAN (/ (* SUMD2 (TAN ANG1)) DIFD)))
                   (SETQ TTN2 (+ (* DIFD (SIN ANG22)) (* SUM3 (SIN ANG1))))
                   (SETQ TTD2 (* 1500 (* (SIN ANG1) (SIN ANG22))))
                   (SETQ SB2_TIME (/ TTN2 TTD2))
                   (SETQ S2B2_RET (APPEND (TRANS_SIG SB2_TIME BCDUNT)
                                          S2B2_RET2))
;
                   (SETQ BCDUNT (- BCDUNT 1))
                   (GD LDDP)))))
;
            (T
              (SETQ S2B2_RIDGE 'ND))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN B2S2_GEN (DEPTB1 VELOC DEPTB2)                                                    ; FUNCTION TO PROMPT FOR AND COMPUTE
;                                                                                        ;    TWO BOTTOM AND TWO SURFACE RETURNS
      (SETQ B2S2_RET ())
      (SETQ B2S2_RET1 ())
      (SETQ B2S2_RET2 ())
      (TERPRI)
      (PRINC "Are bott(2)surf(2) boundary returns desired? [Y/N] ..... ")
      (SETO B2S2_RIDGE (READ))
      (COND ((EOUAL B2S2_RIDGE 'Y)                                                       ; IF DESIRED, GENERATE B2S2 BOUNDARY RETDRNS
;
             (PROG (BCOUNT)
                   (SETQ BCOUNT 31)
;
                   LOOP
;
                   (COND ((OR (EQDAL BCOUNT 0) (<= BCOUNT (- 32 (/ VELOC 2))))
                              (SETQ B2S2_RET (APPEND B2S2_RET1 B2S2_RET2))
                              (RETURN B2S2_RET))
                         (T
;
                   (SETO DELF (* (- 32 BCOUNT) 20))
                   (SETQ ANG1 (ACOS (- 1 (/ DELF (* 10 VELOC)))))
                   (SETQ DIF4 (- (* 4 DEPTB2) DEPTB1))
                   (SETO DIFD (- DEPTB2 DEPTB1))
                   (SETQ ANG2 (ATAN (/ (* DIF4 (TAN ANG1)) DEPTB1)))
                   (SETQ TTN (+ (* DEPTB1 (SIN ANG2)) (* DIF4 (SIN ANG1))))
                   (SETQ TTD (* 1500 (* (SIN ANG1) (SIN ANG2))))
                   (SETQ SB_TIME (/ TTN TTD))
                   (SETQ B2S2_RET1 (APPEND (TRANS_SIG SB_TIME BCOUNT)
                                           B2S2_RET1))
;
                   (SETQ DIF22 (- (* 2 DEPTB2) DEPTB1))
                   (SETO ANG22 (ATAN (/ (* DIF22 (TAN ANG1)) DEPTB1)))
                   (SETO TTN2 (+ (* DEPTB1 (SIN ANG22)) (* DIF4 (SIN ANG1))))
                   (SETO TTD2 (* 1500 (* (SIN ANG1) (SIN ANG22))))
                   (SETO SB2_TIME (/ TTN2 TTD2))
                   (SETQ B2S2_RET2 (APPEND (TRANS_SIG SB2_TIME BCOUNT)
                                           B2S2_RET2))
;
                   (SETQ BCOUNT (- BCOUNT 1))
                   (GO LOOP)))))
;
            (T
              (SETQ B2S2_RIDGE 'NO))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN BOUND_WRITE (BOUND_FILE IF_BOUND_GEN)                                              ; FUNCTION TO WRITE BOUNDARY CONDITION VALDES
;                                                                                         ;    TO FILE BOUND_FILE (OUTPDT FILE CENTERS.LSP)
  (WRITE IF_BOUND_GEN :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BODND_FILE)
;
       (COND ((EQUAL IF_BOUND_GEN 'Y)
;
  (WRITE PLATD            :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE PLATV            :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE WATERD           :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE BOUND_AVG_LEV    :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BODND_FILE)
  (WRITE BOUND_CONS_LEV   :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE SURF_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE SURF_RET         :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE BOTT_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE BOTT_RET         :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE S1B1_RIDGE       :STREAM BOUND_FILE)   (WRITE-CHAR #\NEWLINE BOUND_FILE)
  (WRITE B1S1_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE S2B1_RIDGE       :STREAM BOUND_FILE)   (WRITE-CHAR #\NEWLINE BOUND_FILE)
  (WRITE B2S1_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE S2B2_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE B2S2_RIDGE       :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE)
  (WRITE BOUND_RET        :STREAM BOUND_FILE)   (WRITE-CBAR #\NEWLINE BOUND_FILE))
;
             (T
  (WRITE BOUND_RET       :STREAM BOUND_FILE)    (WRITE-CHAR #\NEWLINE BOUND_FILE))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN BOUND_READ (INPUT_FILE)                                                   ; FUNCTION TO REAO COMPONENTS OF BOUNDARY CONDITIONS
;                                                                                ;    FROM FILE WHEN MAP IS REGENERATED
    (SETQ BOUND_GEN       (READ INPUT_FILE))
;
    (CONO ((EQUAL BOUND_GEN 'Y)
           (SETQ PLATD            (READ INPUT_FILE))
           (SETQ PLATV            (READ INPUT_FILE))
           (SETQ WATERD           (READ INPUT_FILE))
           (SETQ BOUND_AVG_LEV    (READ INPUT_FILE))
           (SETQ BOUND_CONS_LEV   (REAO INPUT_FILE))
           (SETQ SURF_RIDGE       (READ INPUT_FILE))
           (SETQ SURF_RET         (REAO INPUT_FILE))
           (SETQ BOTT_RIDGE       (READ INPUT_FILE))
           (SETQ BOTT_RET         (READ INPUT_FILE))
           (SETQ S1B1_RIDGE       (READ INPUT_FILE))
           (SETQ B1S1_RIDGE       (READ INPUT_FILE))
           (SETQ S2B1_RIDGE       (READ INPUT_FILE))
           (SETQ B2S1_RIDGE       (REAO INPUT_FILE))
           (SETQ S2B2_RIDGE       (READ INPUT_FILE))
           (SETQ B2S2_RIDGE       (READ INPUT_FILE))
           (SETQ BOUND_RET        (READ INPUT_FILE)))
;
          (T
           (SETQ BOUND_RET        (READ INPUT_FILE)))))
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
(DEFUN CALC_BD_VAL (RD_POS RET_LIST IF_GEN IF_CONS_LEV AVG_LEV)                ; FUNCTION TO CALCULATE BOUNDARY RETURNS
;
       (CONO ((AND (EQUAL IF_GEN 'Y)
                   (EQUAL IF_CONS_LEV 'Y)
                   (EQUAL RD_POS (CAR (MEMBER RD_POS RET_LIST))))
;
              (COND ((OR (EQUAL RD_POS (CAR (MEMBER RD_POS SURF_RET)))
                         (EQUAL RD_POS (CAR (MEMBER RD_POS BOTT_RET))))
                     (SETQ BD_VAL AVG_LEV))
                    (T
                     (SETQ BD_VAL (/ AVG_LEV 2)))))
;
             ((AND (EQUAL IF_GEN 'Y)
                   (EQUAL RD_POS (CAR (MEMBER RD_POS RET_LIST))))
;
              (COND ((OR (EQUAL RD_POS (CAR (MEMBER RD_POS SURF_RET)))
                         (EQUAL RD_POS (CAR (MEMBER RD_POS BOTT_RET))))
                     (SETQ BD_VAL (DISTRIBUTE AVG_LEV)))
                    (T
                     (SETQ BO_VAL (OISTRIBUTE (/ AVG_LEV 2))))))
;
             (T
               (SETQ BD_VAL 0)))
;
       (SETQ BD_VAL_SQ (* BD_VAL BD_VAL)))                                      ; SQUARE VALUE OF BD_VAL TO ADD TO AMPLITUDE LEVEL
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

APPENDIX B

SAMPLE INTERACTIVE TERMINAL SESSIONS

APPENDIX B

Sample Interactive Terminal Sessions

```
Method 1 - New range-Doppler map :

Would you like a new (N) range-Doppler map or will this be a second-generation (S) map? [N/S]          n

   Are signals desired on the range Doppler map? [Y/N]                                                 y
       Enter the number of 'fat' L signals desired (MAX 12) ...                                        1
       Enter the number of 'fat' O signals desired (MAX 12) ...                                        1
       Enter the number of 'fat' T signals desired (MAX 12) ...                                        1
       Enter the number of L signals desired (MAX 12) ...                                              1
       Enter the number of O signals desired (MAX 12) ...                                              1
       Enter the number of T signals desired (MAX 12) ...                                              1
       Enter the number of bars desired (MAX 12) ...                                                   0
       Enter the desired mean for the level of these signals (MAX 255) ...                           200
       Are constant level signals (C) or signal level that vary randomly (R) desired? [C/R]            c
   Is noise desired on the range Doppler map? [Y/N]                                                    y
       Enter the desired mean for the level of the noise (MAX 255) ...                                20
   Is reverberation desired on the range Doppler map? [Y/N]                                            y
       Enter the desired mean for the level of the reverberation (MAX 255) ...                       200
       Enter the desired value to determine the shape of the reverberation (MAX 64) ...                5
       Enter the desired mean for the random component of the reverberation (MAX 255) ...             50
   Are boundary returns desired? [Y/N] ...                                                             y
       Platform depth (m) ...                                                                        200
       Platform velocity (m/sec) ...                                                                  60
       Water depth (m) ...                                                                           600
       Average level of all boundary returns ...                                                     100
       Are constant boundary returns desired? [Y/N] ...                                                n
       Are surface boundary returns desired? [Y/N] ...                                                 y
       Are bottom boundary returns desired? [Y/N] ...                                                  y
       Are surf(1)bott(1) boundary returns desired? [Y/N] ...                                          y
       Are bott(1)surf(1) boundary returns desired? [Y/N] ...                                          y
       Are surf(2)bott(1) boundary returns desired? [Y/N] ...                                          y
       Are bott(2)surf(1) boundary returns desired? [Y/N] ...                                          y
       Are surf(2)bott(2) boundary returns desired? [Y/N] ...                                          y
       Are bott(2)surf(2) boundary returns desired? [Y/N] ...                                          y

Is a quantized representation of values (Q) or a representation that shows only the presence of a signal (P) desired? [Q/P]  q

-----------------------------------------------------------------------------------------------------------------------

Method 2 - Second-generation range-Doppler map :

Would you like a new (N) range-Doppler map or will this be a second-generation (S) map? [N/S]          s

   The last range Doppler map had these signals and center values:  ((T 1440) (O 800) (L 1457) (FT 2248) (FO 2006) (FL 1834))

       How many of these signals are to be repositioned?                                               4
       Enter a signal to reposition, as it appears above.                                      (FL 1834)
       Enter the desired time change (sec).                                                           10
       Enter a signal to reposition, as it appears above.                                      (FO 2006)
       Enter the desired time change (sec).                                                           10
       Enter a signal to reposition, as it appears above.                                       (L 1457)
       Enter the desired time change (sec).                                                           10
       Enter a signal to reposition, as it appears above.                                      (FT 2248)
       Enter the desired time change (sec).                                                           10
```

APPENDIX C

SAMPLE OUTPUT FILES

APPENDIX C1

CENTERS.LSP

1	Flag for presence of signals
1	Number of 'fat' L signals
1	Number of 'fat' O signals
1	Number of 'fat' T signals
1	Number of L signals
1	Number of O signals
1	Number of T signals
0	Number of bars
200	Mean level of amplitude for signals
C	Type of signal (constant or variable)
1	Flag for presence of noise
20	Mean level of amplitude for noise
1	Flag for presence of reverberation
200	Mean level of amplitude for reverberation
5	Level for shape of reverberation
50	Mean level of amplitude for background noise
1	Flag for quantified values
Y	Flag for presence of boundary returns
200	Platform depth
60	Platform velocity
600	Water depth
100	Mean level of amplitude for boundary returns
N	Flag for constant boundary returns
Y	Flag for presence of surface returns
(388 324 389 325 390 326 391 327 ...)	Surface ridge values (computed by RENSGEN)
Y	Flag for presence of bottom returns
(708 644 709 645 710 646 711 647 ...)	Bottom ridge values (computed by RENSGEN)
Y	Flag for presence of surface-bottom returns
Y	Flag for presence of bottom-surface returns
Y	Flag for presence of surface-bottom-surface returns
Y	Flag for presence of bottom-surface-bottom returns
Y	Flag for presence of surface-bottom-surface-bottom returns
Y	Flag for presence of bottom-surface-bottom-surface returns
((T 1440) (O 800) (L 1457) (FT 2248) (FO 2006) (FL 1834))	Signals and center values

APPENDIX C2

SIGNALS.LSP

```
This file contains quantized values of each cell generated in RENSGEN.
Each row below corresponds to one row on the range-Doppler map.

"QUANTIZED SIGNAL MEAN :  "7
"QUANTIZED REVERBERATION MEAN LEVEL:  "7
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 1 3 5 4 3 5 4 7 7 7 9 8 7 8 7 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 3 3 5 4 6 8 8 8 9 9 9 7 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 7 7 7 7 7 7 7 7 7 1 1 1 1 1 1 4 2 2 2 3 3 5 7 6 8 9 9 9 8 8 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 1 7 2 2 4 3 4 7 4 5 7 8 8 9 8 8 7 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 1 1 2 2 4 4 5 5 6 6 8 9 9 9 8 6 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 2 5 3 4 5 5 6 6 5 5 8 7 8 7 9 8 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 3 1 1 2 8 3 1 2 3 2 2 2 2 2 4 3 3 4 7 4 4 4 5 7 9 8 9 9 9 8 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 2 1 3 2 4 1 1 1 2 4 2 2 1 2 2 4 2 7 9 7 3 5 4 7 9 8 7 9 9 8 5 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 3 7 8 7 5 3 7 6 7 9 9 8 8 8 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 2 4 3 2 3 7 7 8 5 4 6 8 7 6 7 9 9 6 9 4 1 1 1 1 1 1 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 1 1 2 3 4 7 3 6 5 8 7 7 8 7 9 8 8 5 4 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 2 1 4 2 3 3 2 3 3 5 1 2 3 3 3 6 4 6 5 6 7 7 9 8 9 8 9 6 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 2 2 1 1 1 2 1 2 1 2 2 3 3 1 2 3 2 3 5 3 3 5 4 7 6 8 9 9 9 8 6 5 1 1 1 1 1 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 3 3 2 1 2 1 3 1 1 1 1 1 1 1 3 4 6 2 4 3 6 7 5 6 8 7 9 7 7 5 5 1 1 1 1 1 1 7 7 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 3 3 5 4 3 3 4 4 5 7 7 9 8 9 6 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 4 3 2 3 3 3 3 3 4 6 9 9 9 9 9 9 8 8 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 3 1 1 2 2 3 1 1 2 2 4 3 4 5 5 5 6 6 8 8 8 9 9 8 3 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 3 3 2 6 1 2 3 5 2 2 1 2 1 1 2 3 5 4 6 6 4 4 4 7 8 8 7 9 9 8 6 4 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 5 4 2 3 3 1 3 3 2 3 2 4 5 3 1 3 3 2 5 2 3 5 6 7 7 9 9 7 9 7 7 4 1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 4 2 4 3 5 6 7 5 8 9 9 9 9 6 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 7 7 7 7 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 4 5 3 5 5 6 5 6 9 7 7 8 9 7 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 3 4 3 4 6 4 5 8 7 9 8 8 8 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 2 1 3 3 4 1 4 2 2 2 4 2 2 3 3 3 2 3 3 3 5 5 8 6 7 7 9 8 7 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 3 3 1 2 2 2 3 2 5 3 4 3 3 3 5 2 2 3 3 4 4 8 6 6 7 8 9 7 8 7 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 4 3 3 4 6 6 4 6 6 9 8 9 8 7 5 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 2 7 4 7 7 7 7 7 8 8 8 9 9 9 6 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 4 8 2 3 2 2 6 5 4 5 8 9 9 9 8 9 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 3 4 4 7 3 5 7 3 5 5 3 3 3 4 5 7 6 7 9 9 9 9 5 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 4 4 5 7 5 5 9 5 1 3 2 5 7 1 1 1 3 5 3 3 4 5 6 5 7 7 8 9 9 9 6 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 5 7 7 4 6 4 5 1 1 1 1 1 1 1 1 4 1 3 5 5 3 6 4 5 7 9 9 9 9 8 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 4 3 5 6 8 6 9 9 7 9 9 7 7 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 3 5 3 5 7 7 5 7 8 8 8 6 7 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 4 5 3 5 4 5 7 6 5 6 7 8 8 8 8 9 6 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 3 5 3 4 2 5 6 9 6 5 8 5 8 4 2 6 3 3 3 2 3 5 7 6 7 8 8 9 8 8 7 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 6 4 6 2 1 7 6 6 5 5 3 3 3 1 1 3 2 2 3 4 3 3 4 6 6 7 8 9 9 7 5 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 4 5 4 5 7 7 8 9 7 9 7 6 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 4 2 4 5 4 5 7 8 9 8 8 8 8 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 4 2 3 4 5 5 5 7 9 9 9 8 7 5 6 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 3 4 4 4 5 6 7 8 9 9 8 7 7 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 2 3 2 3 4 5 5 6 9 9 9 8 8 8 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)
```

APPENDIX C3

READIN.LSP

```
This file contains the individual cell values computed in RENSCEN.
The rows below are referenced in RENSCEN as rows 0 through 39 (read from bottom to top).
Each row of the range-Doppler map corresponds to two lines of output in READIN.LSP. For example,
the values in row 39 of the map are contained in the top two rows below, starting with   (52 9 27 ...
and ending with      ... 21 19 8).

*SIGNAL MEAN :  *200
*REVERBERATION MEAN LEVEL :  *200
*FLAT NOISE MEAN :  *20
(52 9 27 48 28 8 27 22 10 33 17 21 12 16 33 18 5 31 39 43 42 60 94 79 159 190 151 198 215 231 234 258)
(222 293 187 167 29 39 33 34 42 18 31 3 17 80 12 18 33 17 17 23 53 26 12 17 8 21 29 11 28 21 19 8)
(34 23 32 37 28 7 14 19 28 33 15 36 34 19 27 25 28 29 18 82 50 105 94 74 105 99 119 203 280 252 289 267)
(274 201 180 135 6 20 34 10 7 23 43 34 8 18 34 14 17 24 16 17 40 23 20 32 14 34 9 6 22 22 25 12)
(28 13 26 22 200 201 201 204 202 200 203 202 201 6 28 32 6 32 23 37 69 29 26 147 96 107 182 169 344 228 216 341)
(222 231 205 96 21 13 24 3 30 28 40 7 41 9 16 34 35 22 2 26 35 32 14 13 32 25 16 37 38 19 47 29)
(13 23 24 28 31 35 41 201 200 200 17 38 49 26 7 7 31 7 51 61 43 92 96 84 159 122 207 224 202 246 283 245)
(300 254 178 158 12 21 9 20 15 18 23 22 11 53 39 21 49 20 21 30 11 33 9 13 18 35 16 28 55 49 15 20)
(9 9 9 18 7 23 5 201 201 202 30 11 16 8 28 18 11 25 39 99 53 53 64 129 146 139 179 145 237 259 351 276)
(298 178 186 115 12 20 14 34 13 16 17 19 13 32 50 42 21 17 19 4 13 30 1 42 45 20 17 54 8 24 37 20)
(27 29 46 31 17 30 30 201 200 200 10 7 4 43 44 58 24 41 73 74 98 58 90 102 110 187 157 202 199 227 251 306)
(278 212 142 151 14 14 8 15 3 6 18 22 27 28 55 16 40 51 37 59 28 21 23 15 8 12 24 50 31 13 22 8)
(37 6 36 32 46 124 77 43 212 83 107 112 20 90 35 48 75 88 76 144 114 108 244 160 116 134 175 209 225 194 225 309)
(295 200 142 197 39 27 27 13 28 40 5 29 12 12 41 21 16 29 22 6 16 9 22 2 25 16 21 33 22 14 25 12)
(46 21 8 19 13 64 68 53 51 41 27 97 48 68 63 59 95 68 55 54 95 214 224 221 74 82 143 217 184 213 231 299)
(308 190 143 114 19 24 36 28 33 34 8 25 24 22 28 49 7 17 12 20 29 24 19 25 21 7 18 19 31 22 25 7)
(21 11 35 15 57 15 20 20 11 26 48 17 20 39 33 10 30 126 144 149 85 208 207 221 79 163 105 141 267 201 284 315)
(267 227 155 123 18 27 39 15 19 29 13 25 34 27 26 18 20 27 41 9 13 51 5 2 34 28 12 7 21 24 25 43)
(27 26 53 31 3 13 11 29 41 8 26 47 25 17 29 94 61 41 114 180 84 218 237 246 161 194 205 176 237 242 262 246)
(235 263 220 95 14 19 16 35 47 11 200 38 16 30 29 23 24 23 19 36 14 15 8 14 23 40 19 17 27 4 11 16)
(21 23 10 38 15 13 21 27 26 14 14 22 28 47 65 75 45 21 27 60 80 137 204 113 177 167 215 144 229 199 227 238)
(273 214 200 166 31 37 41 34 25 203 200 200 42 37 37 31 29 17 15 18 37 12 20 19 27 9 8 9 22 8 17 22)
(49 37 21 10 6 22 16 16 79 71 101 56 40 57 51 81 89 77 37 79 109 89 115 83 125 96 240 184 198 290 251 290)
(264 230 204 115 31 24 22 32 10 200 201 201 28 29 47 53 45 10 23 30 14 22 17 4 21 16 17 33 9 41 16 7)
(34 26 22 12 91 41 81 110 96 87 161 100 88 44 24 83 103 50 111 133 64 110 130 95 67 135 182 163 294 224 227 294)
(234 219 187 160 54 15 23 32 10 201 202 202 25 1 31 39 30 16 50 42 17 15 17 11 9 36 39 26 12 34 37 16)
(12 30 8 27 47 140 28 57 114 60 62 75 34 19 26 23 30 26 25 124 103 77 59 100 87 128 144 171 242 246 315 284)
(283 242 140 84 49 23 12 39 18 18 200 200 200 200 200 202 46 23 15 9 45 25 25 10 41 45 55 34 29 30 28 16)
(31 17 24 36 13 12 25 41 14 17 12 32 26 13 32 20 9 83 58 152 86 61 175 115 122 128 155 155 227 305 281 258)
(276 204 151 70 48 29 34 21 17 17 36 19 63 17 28 6 5 2 17 19 23 18 30 11 31 10 19 18 26 40 26 15)
(16 20 21 36 6 23 18 8 4 34 46 33 30 14 43 93 35 81 41 32 59 64 55 50 142 95 166 182 184 308 326 340)
(352 282 283 217 50 19 18 13 16 34 54 16 19 40 20 16 44 205 19 27 22 44 16 6 12 46 35 16 20 17 19 10)
(44 14 12 24 31 44 32 18 14 4 52 38 39 43 63 78 54 41 22 116 121 152 116 85 183 169 164 270 232 288 210 240)
(354 202 245 89 29 19 10 25 39 16 21 16 29 46 19 22 4 202 56 28 21 16 21 19 32 16 27 13 5 24 27 8)
(43 14 33 23 72 83 33 91 70 30 61 56 83 72 54 30 83 42 20 69 89 69 81 140 114 131 151 211 185 263 219 271)
(287 234 183 97 7 13 28 24 28 16 35 16 53 56 9 19 30 201 37 28 29 12 26 8 33 31 31 24 13 51 24 40)
(17 11 9 6 62 51 49 78 49 77 93 67 55 114 125 54 104 51 81 150 64 108 191 77 189 138 177 154 238 338 244 279)
(414 351 214 160 14 14 6 8 25 46 30 35 35 30 17 39 18 200 17 28 7 33 32 14 46 19 35 36 2 33 44 34)
(21 30 8 32 6 15 19 8 13 29 14 46 32 31 23 6 7 28 10 120 34 23 66 82 97 118 188 237 279 283 314 218)
(339 246 186 89 15 37 15 3 14 31 20 34 36 37 7 17 27 205 201 200 207 23 18 18 22 11 44 24 28 43 39 13)
(17 19 72 27 18 18 58 21 18 20 31 7 8 20 13 11 22 27 34 47 85 96 68 97 111 155 146 147 202 244 241 271)
(233 248 183 79 5 19 23 20 31 22 11 32 40 52 20 25 5 15 31 11 13 34 17 29 31 21 21 9 24 23 15 42)
(25 24 52 24 18 28 22 4 35 22 43 12 21 29 12 1 6 64 47 106 130 90 56 124 144 209 169 203 222 194 234 246)
(285 179 184 98 22 36 10 25 12 10 23 16 48 26 11 53 26 33 15 20 57 12 29 36 40 10 36 18 17 25 38 15)
(33 8 34 18 51 72 29 48 33 49 49 47 84 49 67 129 37 28 38 87 98 145 117 137 67 143 214 137 169 226 255 236)
(271 216 142 97 15 11 9 9 33 43 22 48 19 13 67 29 27 3 25 11 13 10 28 18 17 37 14 23 18 7 21 21)
(41 8 60 28 97 74 50 125 102 40 60 46 121 40 94 28 53 65 69 62 114 175 184 149 111 141 188 199 225 259 301 262)
(227 254 138 137 23 23 21 25 27 2 40 36 12 62 27 39 13 14 38 14 37 21 11 67 17 34 15 68 20 20 29 33)


(28 19 43 10 15 18 19 21 28 50 21 56 26 9 37 8 20 2 5 82 51 96 72 94 156 213 132 197 235 240 245 361)
(247 227 238 78 31 25 25 19 8 5 4 34 33 7 22 32 30 7 8 36 9 22 20 19 16 21 11 13 26 28 16 31)
(13 8 27 31 12 21 11 43 9 18 18 47 43 22 20 15 27 16 30 89 104 187 98 221 217 126 199 140 216 266 240 267)
(284 252 162 96 10 23 18 29 12 30 26 36 32 30 25 33 6 27 24 51 16 13 43 24 12 7 3 25 25 8 12 30)
(42 23 39 27 14 42 20 13 32 18 32 20 14 36 23 54 43 126 234 200 171 217 278 91 166 111 148 176 188 249 262 342)
(279 309 161 172 17 19 13 12 28 19 41 26 29 7 8 17 12 39 18 30 12 16 69 20 32 39 14 13 38 41 28 31)
(24 29 13 32 46 28 11 28 25 13 27 166 52 87 69 64 152 48 98 103 88 69 115 80 120 88 189 169 179 235 321 322)
(225 279 163 85 31 9 21 29 27 8 22 39 6 7 11 13 42 22 29 13 43 50 14 14 29 23 31 29 22 9 18 29)
(16 18 32 55 138 109 235 154 126 113 123 89 96 237 181 40 132 34 38 55 71 69 60 105 120 154 148 136 194 248 246 290)
(229 264 130 129 24 36 22 24 21 67 19 53 14 44 28 6 27 8 19 20 29 6 27 12 19 7 28 20 20 16 37 46)
(16 18 5 31 92 126 78 76 107 115 228 24 30 29 41 63 0 44 15 91 68 72 66 129 175 136 259 142 256 226 233 240)
(334 208 174 168 25 27 25 9 31 27 14 11 66 15 10 23 15 16 13 29 30 24 16 29 49 34 19 47 20 23 38 12)
(19 16 29 34 14 11 47 31 33 2 28 38 4 40 29 24 12 6 40 77 49 79 114 80 99 124 173 198 261 233 284 249)
(305 250 167 86 39 13 17 26 18 27 15 9 19 20 15 5 35 52 12 8 16 15 22 35 19 22 31 8 12 30 27 14)
(43 48 30 23 46 40 30 24 24 26 19 19 18 30 30 17 3 9 29 144 64 39 75 190 253 204 195 183 225 232 266 243)
(279 278 193 170 41 11 15 25 29 16 24 20 48 10 45 12 33 39 21 17 37 65 31 8 9 35 3 14 60 25 39 21)
(53 3 3 15 12 19 25 33 6 5 34 23 25 15 4 21 14 80 214 93 292 216 97 54 142 201 149 180 240 325 262 261)
(238 198 212 131 2 17 11 26 10 19 14 24 45 20 44 18 40 16 32 43 31 30 35 14 7 28 24 12 46 21 34 25)
(26 32 29 34 124 189 65 243 199 42 185 141 188 169 182 38 280 92 157 67 116 86 125 181 107 107 251 199 225 246 244 333)
(269 200 180 153 30 16 34 39 14 19 46 31 21 43 20 25 13 28 4 34 28 32 22 20 13 5 36 9 10 15 14 10)
(31 40 36 32 224 88 84 164 124 247 220 122 169 142 173 117 131 11 42 38 82 79 126 135 133 141 214 151 266 229 264 277)
(287 231 191 151 58 23 10 23 40 3 26 18 45 18 16 19 6 22 8 34 13 11 29 23 33 56 33 10 57 51 31 29)
(18 19 2 23 22 26 24 29 10 13 32 24 13 11 48 40 18 28 24 21 44 87 77 66 103 132 189 187 157 219 239 284)
(204 239 140 134 8 14 12 19 40 23 33 14 14 14 36 27 39 19 20 40 22 3 29 24 16 15 44 27 11 30 37 49)
(19 16 6 4 14 20 28 19 30 10 30 42 47 33 19 43 34 15 85 59 45 57 82 114 155 153 150 221 238 207 268 309)
(278 187 185 134 23 15 28 22 6 37 20 15 42 8 61 53 50 34 35 22 24 18 22 19 10 15 24 37 18 10 22 17)
(9 27 23 8 37 13 23 17 16 9 51 42 41 9 0 33 20 14 32 54 50 56 53 90 119 124 182 159 214 270 223 219)
(256 204 180 113 23 38 6 18 25 26 21 23 52 15 23 22 20 10 7 57 14 23 43 17 14 38 35 26 22 12 37 11)
(12 23 19 17 5 22 23 10 22 41 6 31 16 18 27 7 11 16 31 62 36 111 99 53 120 103 176 218 247 295 296 276)
(294 249 161 73 54 35 36 19 17 38 4 9 47 31 30 17 34 29 24 22 41 14 24 26 41 16 38 6 48 35 32 55)
(49 40 12 12 14 18 14 67 32 24 20 39 47 19 37 10 24 22 55 66 85 82 90 94 106 182 148 170 275 245 271 245)
(281 227 215 99 68 9 32 7 13 40 28 17 17 9 13 18 6 32 62 23 20 6 14 39 19 4 15 12 38 12 26 36)
```

APPENDIX D

THREE-DIMENSIONAL GRAPH PROGRAM

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C                                                                         C
C       FORTRAN program RDPLOT.FOR                                        C
C                                                                         C
C                                       Author:     Joe Wakeley           C
C                                                                         C
C                                       Consultant: Kent Eschenberg       C
C                                                                         C
C                                       Revised:    12 December 1985      C
C                                                                         C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C       RDPLOT.FOR is a program to be used with the output of RENSGEN.LSP, i.e.
C       PLOT3D.LSP.  RDPLOT.FOR creates the necessary PDF file for plotting
C       the Range/Doppler Map generated in REVGEN.LSP and stored in PLOT3D.LSP.
C
C       RDPLOT.FOR is configured to accept multiple lists of numbers
C       representing a Range/Doppler Map generated by the LISP program
C       REVGEN.LSP.  RDPLOT.FOR is setup to accept 40 lists of 64 integer
C       numbers from REVGEN.LSP.
C
C       After compiling RDPLOT, LINKing to the TEMPLATE library is required
C       before RUNning.  For example, after changes have been made to
C       RDPLOT.FOR the following is necessary;
C                                              FOR RDPLOT.FOR <RTN>
C                                              LINK RDPLOT,TEMPLATE/LIB <RTN>
C                                              RUN RDPLOT <RTN>
C
C       As a result of running RDPLOT, screen width set to 132, the user will
C       be prompted for inputs necessary to setup the Range/Doppler Map plot.
C       A file suitable for plotting on a graphics terminal or using the Laser
C       Printer will be be generated named RDPLOT.PDF.  The output of running
C       RDPLOT is RDPLOT.PDF.  Other similar PDF files should be renamed to
C       eliminate having multiple plots in one file.
C
C       RDPLOT.PDF may be viewed on a graphics terminal by entering the
C       following;
C                         POSTPLOT <RTN>
C                         TK4 <RTN>
C                         INPUT 'RDPLOT.PDF' 7. <RTN>
C                         FORMAT MAXIMUM <RTN>
C                         PAGE 1 <RTN>
C
C       If multiple plots are in the same PDF file, and they wished to be all
C       viewed on the terminal, follow the above PAGE 1 <RTN> with;
C                                                             AERASE <RTN>
C                                                             PAGE 2 <RTN>
C                                                             etc
C
C       After viewing the resulting plot the $ prompt may be returned to and
C       the screen cleared by entering the following;
C                                                    EXIT <RTN>
C                                                    CLS <RTN>
C
C       If a Laser Printer copy of the plot is desired enter the following
C       from the $ prompt;
C                               IMPTEMPL RDPLOT.PDF <RTN>
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C
        INTEGER NUMBIX(40,64),NUMBIXR(64,40)
C
        REAL XNUMBR(40,64),XNUMBRR(64,40),WORK1(8000),WORK2(64)
C
        CHARACTER FILE_NAME_IN*50,STRING*360,TITLE*40,TITLE1*60,FILD*1
        CHARACTER VLABEL(2)*20/ 'Relative', 'Amplitude'/,ROTATION*1
C
C
C                                    ask for input data file name
C
        WRITE (6,100)
 100           FORMAT (2X,'Name of input data file          ',$)
        READ (5,110) FILE_NAME_IN
 110            FORMAT (A50)
        OPEN (UNIT=99,FILE=FILE_NAME_IN,STATUS='OLD',CARRIAGE CONTROL='NONE',
     >      READONLY)
C
C                                    identify array and size
C
        NCOL=64
        WRITE (6,140) NCOL
 140          FORMAT (2X,'Number of frequency bins             ',I2)
C
        NROW=40
        WRITE (6,160) NROW
 160          FORMAT (2X,'Number of Fourier transforms         ',I2)
C
        WRITE (6,400)
 400          FORMAT (2X,'Is a 90 deg rotation desired (Y/N)   ',$)
        READ (5,410) ROTATION
 410         FORMAT (A)
        CALL STR$UPCASE(ROTATION,ROTATION)
C
        WRITE (6,420)
 420    FORMAT (2X,'Enter 40 character plot title ending with a  \',$)
        READ (5,410) TITLE
        CALL STR$UPCASE(TITLE,TITLE)
        WRITE (6,430)
 430    FORMAT (2X,'Enter 60 character continuation of title     \',$)
        READ (5,410) TITLE1
        CALL STR$UPCASE(TITLE1,TITLE1)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          measure data line length, and read data
C
      DO J=1,NROW
         READ (99,120) LENGTH, STRING(1:LENGTH)
 120            FORMAT (Q,A)
C
C                          change right and left parentheses to blanks
C
         STRING(1:1)=' '
         STRING(LENGTH:LENGTH)=' '
         READ (STRING,*) (NUMBIX(J,I),I=1,NCOL)
C
         IF(ROTATION.EQ.'Y') THEN
                            DO I=1,NCOL
                               NCOLR=NROW+1-J
                               NUMBIXR(I,NCOLR)=NUMBIX(J,I)
                            END DO
                       END IF
      END DO
      CLOSE (UNIT=99)
C
C                          write output to files
C
      WRITE (6,700)
 700            FORMAT (2X,'Are output data files desired (Y/N)      ',$)
      READ (5,710) FILD
 710           FORMAT (A)
      CALL STR$UPCASE(FILD,FILD)
      IF(FILD.EQ.'Y') THEN
        IF(ROTATION.EQ.'Y') THEN
                            OPEN (UNIT=20,STATUS='NEW')
                            OPEN (UNIT=40,STATUS='NEW')
                            OPEN (UNIT=87,STATUS='NEW')
             WRITE (20,131) ((NUMBIXR(J,I), I=1,NROW-20),J=1,NCOL)
             WRITE (40,131) ((NUMBIXR(J,I), I=NROW-19,NROW),J=1,NCOL)
             WRITE (87,131) ((NUMBIXR(J,I), I=1,NROW),J=1,NCOL)
                            CLOSE (UNIT=20)
                            CLOSE (UNIT=40)
                            CLOSE (UNIT=87)
                        ELSE
                            OPEN (UNIT=32,STATUS='NEW')
                            OPEN (UNIT=64,STATUS='NEW')
                            OPEN (UNIT=88,STATUS='NEW')
             WRITE (32,130) ((NUMBIX(J,I), I=1,NCOL-32),J=1,NROW)
             WRITE (64,130) ((NUMBIX(J,I), I=NCOL-31,NCOL),J=1,NROW)
             WRITE (88,130) ((NUMBIX(J,I), I=1,NCOL),J=1,NROW)
                            CLOSE (UNIT=32)
                            CLOSE (UNIT=64)
                            CLOSE (UNIT=88)
                        END IF
      END IF
 130  FORMAT (32(1X,I3))
 131  FORMAT (20(1X,I3))
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          initialize TEMPLATE and page lay out
C
      CALL USLPDF
      CALL UPSET( 'OUTPUTFILE', 7. )          ! output goes to 7 ...
      CALL UASSGN( 7., 'RDPLOT.PDF\' )        ! ... which goes to this file
      CALL USTART
C
      CALL UDIMEN( 9., 6.5 )                  ! 1 inch borders on 8.5x11 paper
      CALL UPSET( 'WIDTH', 10. )              ! make the pen thick ...
      CALL UOUTLN                             ! ... then draw a border
      CALL ARL_HIDE_FONT                      ! setup text for ARL_HIDE
C
C                          produce a two line plot title
C
      CALL USET('DEVICE')                     ! setup for placement in inches
      CALL UPSET('HORIZONTAL_SIZE',0.28)      ! title horizontal letter size
      CALL UPSET('VERTICAL_SIZE',0.40)        ! title vertical letter size
      CALL UPSET('WIDTH',5.0)                 ! title letter width
      CALL USET('CJUSTIFY')                   ! center title
      CALL UPRINT(4.5,6.0,%REF(TITLE))        ! locate title center on page
      CALL UPSET('HORIZONTAL_SIZE',0.21)      ! title1 horizontal letter size
      CALL UPSET('VERTICAL_SIZE',0.30)        ! title1 vertical letter size
      CALL UPSET('WIDTH',4.0)                 ! title1 letter width
      CALL UPRINT(4.5,5.5,%REF(TITLE1))       ! locate title1 center on page
      CALL USET('VIRTUAL')                    ! reset for plotting
      CALL ARL_HIDE_FONT                      ! reset text for ARL_HIDE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          data array from RENSGEN.LSP called PLOT3D.LSP
      ZMAX=-2000.0
      ZMIN=2000.0
C
      IF (ROTATION.EQ.'Y') THEN
                          DO I=1,NCOL
                            DO J=1,NROW
C
                              XNUMBRR(I,J)=NUMBIXR(I,J)
C
                              ZMAX=MAX(ZMAX,XNUMBRR(I,J))
                              ZMIN=MIN(ZMIN,XNUMBRR(I,J))
C
                            END DO
                          END DO
                       ELSE
                          DO I=1,NROW
                            DO J=1,NCOL
C
                              XNUMBR(I,J)=NUMBIX(I,J)
C
                              ZMAX=MAX(ZMAX,XNUMBR(I,J))
                              ZMIN=MIN(ZMIN,XNUMBR(I,J))
C
                            END DO
                          END DO
                       END IF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                          vertical axis set-up
C
      WRITE (6,210) ZMAX
 210  FORMAT (2X,'Maximum value of the Range-Doppler array is  ',F5.0)
      WRITE (6,220) ZMIN
 220  FORMAT (2X,'Minimum value of the Range-Doppler array is  ',F5.0)
C
      WRITE (6,230)
 230  FORMAT (2X,'Minimum desired value for vertical axis (int) ',$)
      READ (5,250) IVORG
      VMIN=IVORG
 250  FORMAT (I4)
      WRITE (6,240)
 240  FORMAT (2X,'Maximum desired value for vertical axis (int) ',$)
      READ (5,250) IVLIM
      VMAX=IVLIM
      WRITE (6,270)
 270  FORMAT (2X,'Increment desired for vertical axis (int)     ',$)
      READ (5,250) IVINC
      WRITE (6,290)
 290  FORMAT (2X,'Lower cutoff desired for vertical axis (int)  ',$)
      READ (5,250) ICUTOFF
      CUTOFF=FLOAT(ICUTOFF)+0.5
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C                            produce the plot
C
      IF (ROTATION.EQ.'Y') THEN
C
      CALL ARL_HIDE_SCALE( NCOL, NCOL, NROW, 8000, 64,
     >                     VMIN, VMAX, 'CUTON', CUTOFF,
     >                     0.5, 1.0, 1.25, 0.5, 0.5, 0.0,
     >                     45.0, 45.0, 'FDRAW', 'SDRAW' )
C
      CALL ARL_HIDE( XNUMBRR, WORK1, WORK2)
C
      CALL ARL_HIDE_FAXIS( 'SIDE', 'INTEGER',
     >                     0, NCOL, 8, '(I3)',
     >                     'Frequency Bin', 1, 40.0, 20.0 )
C
      CALL ARL_HIDE_SAXIS( 'SIDE', 'INTEGER',
     >                     0, NROW, 5, '(I3)',
     >                     'Fourier Transform Number', 1, 60.0, 20.0 )
C
      CALL ARL_HIDE_VAXIS( 'DUMMY', 'INTEGER',
     >                     IVORG, IVLIM, IVINC, '(I4)',
     >                     vlabel, 2, 50.0, 30.0 )
C
C
                ELSE
C
C
      CALL ARL_HIDE_SCALE( NROW, NROW, NCOL, 8000, 64,
     >                     VMIN, VMAX, 'CUTON', CUTOFF,
     >                     0.5, 1.0, 1.25, 0.5, 0.5, 0.0,
     >                     45.0, 45.0, 'FDRAW', 'SDRAW' )
C
      CALL ARL_HIDE( XNUMBR, WORK1, WORK2)
C
      CALL ARL_HIDE_FAXIS( 'SIDE', 'INTEGER',
     >                     0, NROW, -5, '(I3)',
     >                     'Fourier Transform Number', 1, 40.0, 20.0 )
C
      CALL ARL_HIDE_SAXIS( 'SIDE', 'INTEGER',
     >                     0, NCOL, 8, '(I3)',
     >                     'Frequency Bin', 1, 60.0, 20.0 )
C
      CALL ARL_HIDE_VAXIS( 'DUMMY', 'INTEGER',
     >                     IVORG, IVLIM, IVINC, '(I4)',
     >                     vlabel, 2, 50.0, 35.0 )
C
                END IF
C
      CALL UEND
      STOP
      END
```

APPENDIX E

INSTRUCTIONS FOR USING THE VAX/LISP

APPENDIX E
Instructions for using VAX/LISP

The VAX/LISP may be entered, from the dollar sign prompt, by typing LISP and then <RTN>. A prompt will appear LISP>.

NOTE: (1) Functions may be entered at this point by the operator to become familiar with the VAX/LISP dialect. However, it should be noted that functions entered at this time will be destroyed when the operator exits the VAX/LISP.

(2) To create a permanent record of desired functions the EDT editor may be used prior to entering LISP to create a file identified by FILENAME.LSP and loaded as mentioned below. This should provide a permanent record of the desired files. If the LISP editor is used (entered through the command (ED "RENSGEN.LSP")), a permanent record of the file will be provided by the following:

```
     Enter GOLD COMMAND and type EXIT.
     A prompt will appear, 'EXITING THE EDITOR.
       ALL BUFFERS WILL BE LOST. ENTER [Y]
       TO CONTINUE:'
     Enter a Y.
     The message 'EXITING...' appears.
```

If any changes have been made, a prompt appears 'BUFFER RENSGEN.LSP IS MODIFIED. DO YOU WANT ITS CONTENTS SAVED [Y]:'.

```
     Enter Y to provide a record of changes.
     Messages of 'WRITING FILE RENSGEN.LSP' and
       'WROTE FILE ... RECORDS' will be followed
        by the prompt LISP>.
```

RENSGEN.LSP may then be entered into the VAX/LISP by typing (LOAD "RENSGEN.LSP") and then pressing <RTN>. A message 'LOADING CONTENTS OF FILE RENSGEN.LSP' will appear, all the functions within the program will be sequentially listed, a message 'FINISHED LOADING' and a T will appear if the program loaded correctly, and then a LISP> prompt is given.

NOTE: At this point any function in RENSGEN.LSP may be exercised by typing the function name and appropriate argument(s) enclosed in parentheses, and then <RTN>.

The program RENSGEN.LSP may be exercised by typing (REVERBERATION) after the LISP> prompt following the loading of the program, and answering the questions that follow. To get out of VAX/LISP type (EXIT) after the LISP> prompt. The dollar sign prompt will then appear.

If the program has been compiled, follow the same procedure to load and run, but replace RENSGEN.LSP with RENSGEN.FAS. To compile a LISP program, enter LISP/COMPILE RENSGEN.LSP after the dollar sign prompt.

APPENDIX F

INTERESTING LISP FUNCTIONS

APPENDIX F
Interesting LISP Functions

FUNCALL - (FUNCALL fn a1 a2 ... an) applies the function (fn) to the arguments a1, a2, ... an as if the function was called directly (i.e., (fn a1 a2 ... an)). The FUNCALL function is useful when it is necessary to pass a function name as an argument to another function. The function may not be a special form or macro. An example of the use of FUNCALL can be found in the UP-DATE function of RENSGEN.

GENSYM - GENSYM invents a print name and creates a new symbol with that print name. The invented print name consists of a prefix and a decimal representation of a number. The number is incremented on each call to the GENSYM function.

```
Example:
(gensym 'num) --> num1
(gensym)      --> num2
(gensym 5)    --> num5
(gensym 'new) --> new6
```

GENSYM is used in the INITIAL function of RENSGEN.

PROG - A PROG construct establishes a local environment in a particular function. The variables x1, x2, ... xn below are all local variables.

```
(PROG (x1 x2 ...xn)
      (setq x1 1 x2 2 ... xn n)
 label1
       (cond (.... (return))
       .....
       (t (go label2)))
 label2
       (cond (.... (return))
       ....
       (t (go label1))))
```

Examples of the PROG are found in many functions in RENSGEN, including INITIAL and GENERATE.

LOOP - (LOOP form1 form2 ... formn)
Each form of a LOOP is evaluated in turn. When formn has been evaluated, then form1 is evaluated again, and so on, repeatedly. The LOOP construct never returns a value. Its execution must be terminated explicitly, using a RETURN or THROW, for example. LOOPs in RENSGEN occur in the INITIAL and GENERATE functions.

File reading - To read information from a file, the file must first be opened:

```
(make-pathname :version :newest)
(setq INPUTFILE (open "FILENAME.FILETYPE"))
```

Then to set a variable (var) to a value from the file:

```
(setq var (read INPUTFILE))
```

When all information has been read from the file, the file is closed:

```
(close INPUTFILE)
```

The REGEN function contains examples of file reading in RENSGEN.

File writing - To write to a file, again the file must be opened:

```
(setq OUTPUTFILE (open "FN.FT;1"
   :direction :output :if-exists :new-version))
```

One of the functions in RENSGEN to write to files is the REGEN function.

Compiling - To compile a LISP program, the command

```
LISP/COMPILE FN.LSP
```

can be issued after the dollar sign prompt. If the LISP editor is being used, the command is

```
(compile "FILENAME.LSP").
```

(NOTE: A LISP file must have the filetype LSP.)

April 18, 1986
JES:JW:mdp

APPENDIX G

DATA FOR GENERATION OF FIGURES

APPENDIX G

Data for Generation of Figures

Figure 1 :		
	Signal Types	- None
	Noise Mean	- 20
	Reverberation Mean	- 200
	Reverberation Shape	- 3
	Reverberation Noise	- 50
	Platform Depth	- 200 (m)
	Platform Velocity	- 60 (m/sec)
	Water Depth	- 600 (m)
	Boundary Return Mean	- 100
	Cutoff for Graph	- 35 Variable
Figure 2:	Measured Data	
Figure 3:	Signal Types	- None
	Reverberation Mean	- 200
	Reverberation Shape	- 3
	Reverberation Noise	- 50
	Platform Depth	- 200 (m)
	Platform Velocity	- 60 (m/sec)
	Water Depth	- 450 (m)
	Boundary Return Mean	- 75 Variable
	Cutoff for Graph	- 25
Figure 4:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
Figure 5:	Signal Types	- L,FL,O,FO,T,FT,Bar
	Signal Mean	- 200 Constant
Figure 6:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Variable
Figure 7:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
	Noise Mean	- 20
Figure 8:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
	Noise Mean	- 50
Figure 9:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
	Reverberation	- 200
	Reverberation Shape	- 5
	Reverberation Noise	- 50

Figure 10:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
	Reverberation	- 200
	Reverberation Shape	- 5
	Reverberation Noise	- 50
	Platform Depth	- 200 (m)
	Platform Velocity	- 60 (m/sec)
	Water Depth	- 600 (m)
	Boundary Return Mean	- 100 Variable
Figure 11:	Signal Types	- L,FL,O,FO,T,FT
	Signal Mean	- 200 Constant
	Noise Mean	- 20
	Reverberation	- 200
	Reverberation Shape	- 5
	Reverberation Noise	- 50
	Platform Depth	- 200 (m)
	Platform Velocity	- 60 (m/sec)
	Water Depth	- 600 (m)
	Boundary Return Mean	- 100 Variable

Figure 12: Data same as Figure 11 with 90 Degree Rotation of Horizontal Plane about a Vertical Axis

Figure 13:	Signal Types	- L,FL,O,FO,T,FT
	Repositioned Signals	- L,FL,FO,FT
	Change in Time	- 10 (sec)

APPENDIX H

REFERENCES

APPENDIX H

References

1) Wilensky, Robert, "LISPCRAFT," University of California, Berkeley, W. W. Norton and Company, 1984.

2) Winston, Patrick, H. and Horn, Berthold K. P., "LISP," Massachusetts Institute of Technology, Addison-Wesley Publishing Company, 1981.

3) Steele, Guy L., "Common LISP - The Language," Digital Press, 1984.

DISTRIBUTION LIST FOR UNCLASSIFIED TN 86-64, by J. E. Sentz and J. Wakeley, dated 18 April 1986

Office of Naval Technology (ONT)
800 N. Quincy Street
Arlington, VA 22217-5000
Attention: A. J. Faulstitch, OCNR 23
Copy 1

Office of Naval Technology (ONT)
800 N. Quincy Street
Arlington, VA 22217-5000
Attention: D. C. Houser, OCNR 232
Copy 2

Commander
Naval Ocean Systems Center (NOSC)
San Diego, CA 92152-5000
Attention: William Tagney, Code 613
Copy 3

Commander
Naval Ocean Systems Center (NOSC)
San Diego, CA 92152-5000
Attention: Paul Reeves, Code 632
Copy 4

Commander
Naval Sea Systems Command (NAVSEA)
Department of the Navy
Washington, DC 20362-5101
Attention: Walter Rankin, Code 63D
Copy 5

Commanding Officer
Naval Coastal Systems Center (NCSC)
Panama City, FL 32407-5000
Attention: Paul Kurtz, Code 40
Copy 6

Naval Underwater Systems Center (NUSC)
Department of the Navy
Newport, RI 02841-5047
Attention: C. Albanese, Code 8212
Copy 7

Chris Eggen
Applied Physics Laboratory
University of Washington
1013 N.E. 40th St.
Seattle, WA 98195
Copy 8

DISTRIBUTION LIST FOR UNCLASSIFIED TN 86-64, by J. E. Sentz and J. Wakeley, dated 18 April 1986

Dr. Anthony S. Maida
333 Whitmore Lab
The Pennsylvania State University
University Park, PA 16802
Copy 9

J. E. Sentz
5672-88 Stevens Forest Rd.
Columbia, MD 21045
Copy 10